

Regulated Reflection for Efficient Reasoning in Large reasoning language models

Mingqu Wang* Xinjie Xu Yinan Zou Qi Ye*

School of Information Science and Engineering,
East China University of Science and Technology, Shanghai, China
y30241044@mail.ecust.edu.cn yeh_qi1125@ecust.edu.cn

*Corresponding author.

Abstract

Large reasoning language models (LRLMs) achieve strong performance on complex reasoning tasks by generating long chain-of-thought trajectories, but such long-form reasoning also increases token consumption and latency. Existing inference-time methods that do not fine-tune the target LRLM optimize long reasoning through budgeting, early exit, uncertainty control, or reflection suppression, but they often treat reflection as a local signal rather than as a factor shaping the reasoning trajectory. In this paper, we identify *unregulated reflection* as a source of inefficient reasoning and characterize two failure modes: *Trajectory Drift*, where intermediate reflection opens weakly related branches that restart, reorient, or detour from the current path, and *Verification Saturation*, where answer-adjacent reflection repeatedly checks or restates an already-emerged provisional answer. To address these issues, we propose Regulated Reflection Decoding (RRD), an inference-time framework that requires no fine-tuning or parameter updates to the target LRLM and treats reflection events as online control points. RRD combines Trajectory Drift Suppression (TDS), which guides intermediate reasoning toward validated continuation branches, and Redundant Verification Suppression (RVS), which suppresses redundant verification when the provisional answer is sufficiently reliable. Experiments on multiple reasoning benchmarks show that RRD reduces reasoning length while preserving or improving accuracy, yielding a stronger accuracy–efficiency trade-off than direct inference and representative inference-time baselines.

1 Introduction

Large reasoning language models (LRLMs), represented by OpenAI o1 (OpenAI, 2024a,b) and DeepSeek-R1 (Guo et al., 2025), achieve strong performance on complex reasoning tasks by generating long chain-of-thought (CoT) trajectories before final answers (Wei et al., 2022; Snell et al.,

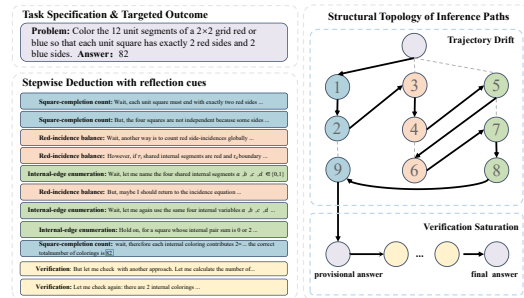


Figure 1: Illustration of two forms of unregulated reflection in LRLM reasoning. The left panel shows the problem and reflective reasoning steps, while the right panel visualizes the corresponding inference paths.

2024). These slow-thinking processes support problem decomposition, intermediate verification, and alternative solution exploration (Zhu and Li, 2025; Wang et al., 2022; Yao et al., 2023), yielding strong results on mathematics (Cobbe et al., 2021; Lightman et al., 2024), programming (Chen et al., 2021; Zhuo et al., 2025), and scientific reasoning benchmarks (Rein et al., 2023). However, long-form reasoning increases token consumption and latency (Sun et al., 2025a; Wang et al., 2025b), and redundant continuation gives rise to overthinking (Chen et al., 2025; Sui et al., 2025). We focus on inference-time control without fine-tuning or modifying the target LRLM. RRD nevertheless uses a separately trained auxiliary continuation validator for TDS.

Training-free inference-time methods are practical for deployed LRLMs because they improve reasoning efficiency without target-model fine-tuning or parameter updates (Qu et al., 2025; Feng et al., 2025). They control reasoning using diverse online signals, including query difficulty, path quality, answer reliability, and local uncertainty. Budgeting and routing methods allocate computation by question complexity (Muennighoff et al., 2025; Han et al., 2025; Ong et al., 2024); reward- and verifier-

guided methods prune low-quality paths (Liao et al., 2025; Lin et al., 2025); confidence-, certainty-, and consistency-based methods stop or reschedule reasoning when the answer or reasoning state appears reliable (Fu et al., 2024; Yang et al., 2025b; Li et al., 2024); entropy-based methods intervene at uncertain tokens or steps (Laaouach, 2025; Yang et al., 2025c); and reflection-oriented methods suppress or stop self-checking, backtracking, strategy switching, and verification (Wang et al., 2025a; Huang et al., 2026; Sun et al., 2025b). Together, these methods show that long reasoning can be optimized at inference time, but they mostly treat reflection as a signal for suppression or stopping rather than as a factor shaping the reasoning trajectory. This motivates our examination of unregulated reflection during inference.

As shown in Figure 1, we identify two forms of unregulated reflection in LRLMs: *Trajectory Drift* and *Verification Saturation*. *Trajectory Drift* occurs during intermediate reasoning, where a reflection event opens auxiliary branches that weakly extend the current path and causes the model to restart, reorient, or pursue unnecessary detours. *Verification Saturation* occurs near answer formation, where reflection mainly repeats checking, revalidating, or restating an already-emerged provisional answer without adding useful reasoning information. These two forms of unregulated reflection highlight that reflection should not be treated only as a token-level pattern (Wang et al., 2025a), an early-exit opportunity (Yang et al., 2025b), or a generic signal for suppression (Huang et al., 2026).

To address them, we propose Regulated Reflection Decoding (RRD), an inference-time framework that requires no fine-tuning or parameter updates to the target LRLM that treats reflection events as online control points. RRD introduces two complementary mechanisms: Trajectory Drift Suppression (TDS), which keeps the model on its current reasoning path when reflection would lead to unnecessary detours, and Redundant Verification Suppression (RVS), which suppresses repeated answer-adjacent checking once the provisional answer is sufficiently reliable. By selectively regulating reflection, RRD reduces generation length while preserving or improving answer accuracy, and extensive experiments show that it also achieves a stronger overall accuracy–efficiency trade-off than representative inference-time baselines.

Our contributions are summarized as follows:

- We identify *unregulated reflection* as a source of inefficient reasoning and characterize two failure modes: *Trajectory Drift* and *Verification Saturation*.
- We propose Regulated Reflection Decoding (RRD), an inference-time framework requiring no target-LRLM fine-tuning that suppresses trajectory drift and redundant verification.
- Experiments on multiple reasoning benchmarks and LRLMs show that RRD improves the accuracy–efficiency trade-off over direct inference and representative baselines.

2 Related Work

Efficient Reasoning for Large Reasoning Models. Long CoT reasoning substantially improves the performance of LRLMs on complex tasks such as mathematics, coding, and scientific reasoning (Pan et al., 2025; Li et al., 2025b). However, the extended reasoning process also brings high inference cost, latency, and redundant continuation, commonly known as overthinking (Cuadron et al., 2025; Fan et al., 2025). Efficient reasoning aims to reduce unnecessary reasoning while preserving final-answer accuracy.

Existing work can be broadly organized into model-based, reasoning-output-based, and input-prompt-based methods (Sui et al., 2025). Model-based methods improve intrinsic reasoning efficiency through length-aware optimization (Aggarwal and Welleck, 2025), concise CoT supervision (Ma et al., 2025b), or model adaptation (Munkhbat et al., 2025), but often require additional training or parameter updates. Input-prompt-based methods control reasoning with concise instructions (Xu et al., 2025), no-thinking prompts (Ma et al., 2025a), or routing strategies (Liang et al., 2025), but rely on instruction following or reliable difficulty estimation. Reasoning-output-based methods instead intervene during generation, making them especially relevant for deployment-time efficiency because they can reduce reasoning cost without modifying model parameters (Jiang et al., 2025; Li et al., 2025a).

Inference-Time Control of Long Reasoning. Inference-time methods reduce reasoning cost by exploiting online signals during generation, including reward, confidence, certainty, consistency, and

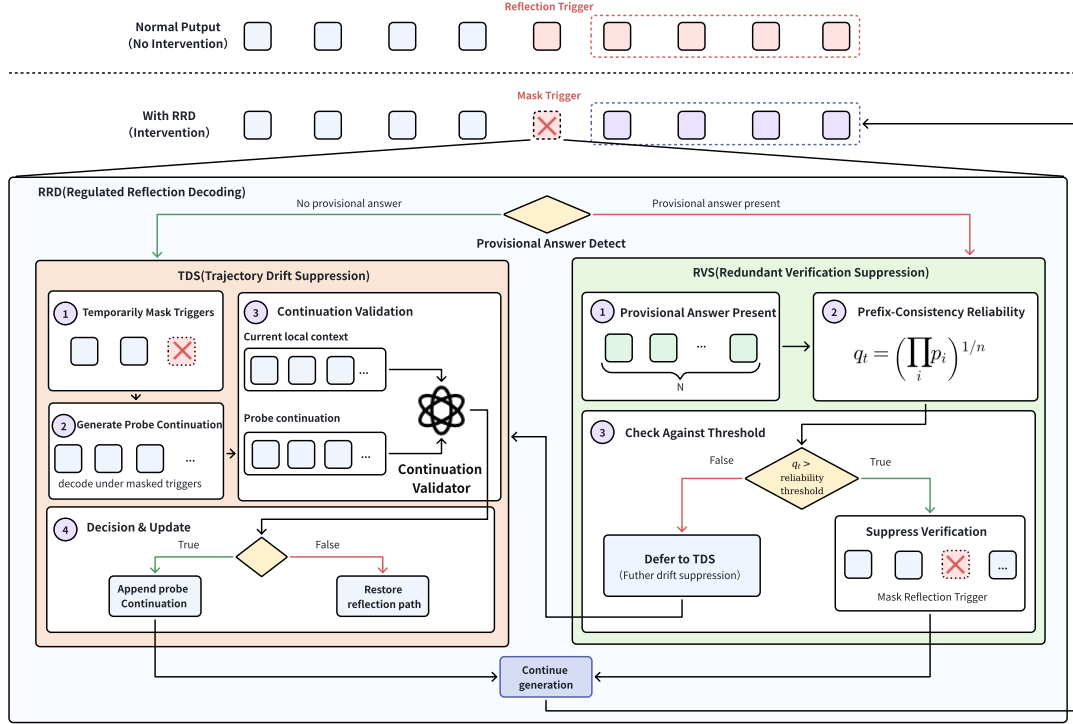


Figure 2: Overview of Regulated Reflection Decoding (RRD). The top compares normal generation with RRD intervention after a reflection trigger, and the bottom illustrates how reflection events are routed to TDS or RVS. Squares denote tokens, diamonds denote decision points, and dashed boxes mark reflection paths or regulated continuations.

entropy. Reward- or verifier-guided methods prune low-quality paths (Lin et al., 2025) or prioritize promising branches (Liao et al., 2025); confidence-, certainty-, and consistency-based methods assess answer likelihood (Fu et al., 2025), distribution sharpness, and cross-path agreement (Liu and Wang, 2025) to stop, truncate, or adapt reasoning when the reasoning state becomes reliable; and entropy-based methods intervene locally at uncertain tokens (Yang et al., 2025c) or steps (Laouach, 2025). A closely related line focuses on reflection behaviors, where LRLMs self-check, backtrack, switch strategies (Zhang et al., 2025), explore alternatives, and verify answers during slow thinking (Ding et al., 2025). NOWAIT suppresses explicit self-reflection tokens (Wang et al., 2025a), CGRS suppresses reflection triggers by certainty (Huang et al., 2026), DEER uses transition points for confidence-based early exit (Yang et al., 2025b), and REFRAIN and ConCISE exploit reflective redundancy (Sun et al., 2025b) or confidence-guided compression (Qiao et al., 2025). These studies demonstrate the usefulness of online signals for controlling long reasoning, but they mainly treat reflection as a token-level suppres-

sion target, a stopping opportunity, or a redundancy pattern, leaving underexplored how unregulated reflection changes the reasoning trajectory itself.

3 Method

3.1 Preliminary

We model the slow-thinking process of an LRLM as an autoregressive reasoning trajectory. Within this trajectory, the model may perform reflection behaviors such as intermediate checking, revision, backtracking, and alternative exploration. Following prior work, we use explicit reflection triggers as observable markers of these behaviors and denote the trigger set as

$$\mathcal{R} = \{\text{Wait, But, Alternatively, ...}\}. \quad (1)$$

The complete trigger set used in our implementation is provided in Appendix C. A reflection event occurs when the model is about to generate a token in $\mathcal{R}$. These events serve as online control points for our method, without requiring offline segmentation of the full reasoning trace.

As illustrated in Figure 2, we propose Regulated Reflection Decoding (RRD), a inference-time

framework that intervenes at reflection events and selectively suppresses inefficient reflection continuations. RRD consists of two mechanisms: Trajectory Drift Suppression (TDS) and Redundant Verification Suppression (RVS). We detail these mechanisms in the following sections.

3.2 Trajectory Drift Suppression

TDS is the first mechanism of RRD and targets trajectory drift during intermediate reasoning. At a reflection event, following the upcoming trigger may shift the model away from the current reasoning trajectory. TDS therefore tests whether suppressing the trigger can still produce a valid continuation of the original reasoning path.

Let ℓ_t denote the original logits at decoding step t , and let x_t be the token selected by the base decoder. TDS is activated only when the selected token is a reflection trigger, i.e., $x_t \in \mathcal{R}$. TDS temporarily masks the trigger set $\mathcal{R}$ in the logits, resulting in the masked logits:

$$\tilde{\ell}_t(v) = \begin{cases} -\infty, & \text{if } v \in \mathcal{R}, \\ \ell_t(v), & \text{otherwise.} \end{cases} \quad (2)$$

A replacement token is then sampled from the masked distribution,

$$\tilde{x}_t \sim \text{softmax}(\tilde{\ell}_t), \quad (3)$$

and the probing branch continues decoding from $\tilde{x}_t$ for a bounded length. This yields a probe continuation

$$u_t = (\tilde{x}_t, u_t^{(2)}, \dots, u_t^{(m)}), \quad (4)$$

where m is the maximum probing length.

TDS then validates whether the probe continuation can extend the original reasoning path under the current local context. Given the current local reasoning context c_t before the reflection event and the probe continuation u_t , the continuation validator predicts

$$D(c_t, u_t) \rightarrow \{\text{CONTINUE}, \text{REFLECT}\}. \quad (5)$$

The label CONTINUE means that u_t can be used as a valid continuation of the original reasoning path under c_t , in which case TDS appends u_t to the main decoding sequence and continues generation from its end. The label REFLECT means that suppressing the trigger fails to produce a valid continuation, in which case TDS discards u_t and restores the

original reflection path. Thus, TDS suppresses trajectory drift only when trigger suppression yields a validated continuation, while preserving reflection otherwise.

The continuation validator is trained on DeepScaleR-derived LRLM reasoning traces with labeled context-continuation pairs; details are provided in Appendix D.

3.3 Redundant Verification Suppression

RVS targets Verification Saturation near answer formation. Once a provisional answer appears in the current prefix, subsequent reflection may mainly repeat checking, revalidating, or restating the same result. RVS is therefore applied only after such a provisional answer has appeared. We identify this condition using lightweight character-level rules, with implementation details provided in Appendix F.

To estimate whether the provisional answer is reliable enough to make further verification redundant, RVS probes an answer from the current reasoning state. Specifically, we append an answer-inducing prompt $\mathcal{I}$, instantiated as Final Answer is, to the current prefix and let the model generate an answer candidate

$$a_t = (a_t^{(1)}, a_t^{(2)}, \dots, a_t^{(n)}), \quad (6)$$

where $a_t^{(i)}$ denotes the i -th token of the generated answer candidate. This probing is used only for reliability estimation and is not inserted into the main decoding sequence. We define the Prefix-Consistency Reliability as

$$q_t = \left(\prod_{i=1}^n p_\theta(a_t^{(i)} \mid x, o_{\leq t}, \mathcal{I}, a_t^{(<i)}) \right)^{1/n}, \quad (7)$$

where x is the input problem, $o_{\leq t}$ is the current reasoning prefix, and $a_t^{(<i)}$ denotes the preceding answer tokens. A larger q_t indicates that the probed answer can be more reliably generated under the current prefix, while a smaller q_t suggests that further checking or revision may still be useful.

Given a reliability threshold τ_a , RVS suppresses the upcoming answer-adjacent reflection trigger only when the probed answer is sufficiently reliable:

$$q_t \geq \tau_a. \quad (8)$$

If this condition is satisfied, RVS applies the same reflection-trigger masking operation as in Eq. 2 at

the current decoding step. Otherwise, RVS preserves the reflection trigger and passes the reflection event to TDS for trajectory-drift checking.

4 Experiments

4.1 Experimental Setup

Datasets. Our evaluation covers four publicly available reasoning benchmarks: AIME24, AIME25, MATH500, and GPQA-Diamond. AIME24 and AIME25 (Art of Problem Solving, 2024, 2025) contain challenging competition-level mathematics problems. MATH500 (Lightman et al., 2024) covers diverse mathematical topics and multi-step reasoning patterns. GPQA-Diamond (Rein et al., 2023) evaluates expert-level scientific reasoning. AIME24 and AIME25 each contain 30 problems, MATH500 contains 500 problems, and GPQA-Diamond contains 198 questions.

Model selection. We conduct experiments on open-source LRLMs spanning different model families and scales. The evaluated models include the Qwen3 family, namely Qwen3-1.7B, Qwen3-8B, and Qwen3-14B (Yang et al., 2025a), and the DeepSeek-R1-Distill series, namely DeepSeek-R1-Distill-Qwen-7B and DeepSeek-R1-Distill-Llama-8B (Guo et al., 2025). This model suite allows us to assess the robustness of RRD across heterogeneous LRLMs.

Baselines. RRD is compared with four representative baselines. Direct Inference (DI) performs standard decoding without intervention. DEER (Yang et al., 2025b) terminates further reasoning through confidence-based early exit. CGRS (Huang et al., 2026) suppresses reflection triggers based on answer-side certainty. MTI (Yang et al., 2025c) applies selective test-time intervention to uncertain tokens.

Evaluation metrics. We report Accuracy (Acc), final-output length (Len), and length reduction (LR) in the main results table. Acc measures final-answer correctness, Len measures the average number of tokens retained in the final output, and LR measures the percentage reduction in final-output tokens relative to Direct Inference. Discarded auxiliary probe tokens are excluded from Len and are accounted for separately, together with total accounted tokens and wall-clock latency, in Table 2. Higher Acc and LR are better, while lower Len indicates higher generation efficiency.

Implementation details. All experiments are im-

plemented with vLLM (Kwon et al., 2023) on NVIDIA GeForce RTX 4090 GPUs. We use temperature 0.6, top- p 0.9, and a maximum context length of 24K tokens. Results are averaged over 5 trials for AIME24/AIME25 and 3 trials for MATH500/GPQA-Diamond. For TDS, the local context length and maximum probe length are set to 100 tokens and 20 tokens respectively. For RVS, the maximum answer-probe length is 20 tokens, and a unified reliability threshold $\tau_a = 0.95$ is used across all models and benchmarks. All method-specific hyperparameters are fixed across experimental settings.

4.2 Main Results

Table 1 reports the main accuracy–output-length results across five reasoning models and four benchmarks. We evaluate each method using answer accuracy (Acc), final-output length (Len), and length reduction (LR) relative to Direct Inference (DI). Averaged over the five model groups, DI achieves an average accuracy of 61.00%, while RRD improves it to 62.08% and reduces final-output length by 25.37% on average. This indicates that RRD substantially shortens the final reasoning output while preserving, and slightly improving, final-answer correctness.

This advantage becomes clearer when compared with existing test-time reasoning baselines. RRD achieves the highest average accuracy among all methods, 62.08%, together with the largest average LR, 25.37%. DEER achieves strong final-output compression with an average LR of 24.18%, but its average accuracy drops to 57.05%, suggesting that reasoning-transition-based early exit may remove useful reasoning together with redundant continuation. CGRS obtains a moderate average LR of 21.93%, but its average accuracy decreases to 59.11%, indicating that suppressing reflection triggers based only on answer-side certainty is insufficient to reliably separate redundant reflection from useful self-correction. MTI, as a token-level test-time intervention baseline, shows a different behavior: it preserves accuracy relatively well, reaching 61.93%, but increases final-output length overall, leading to a negative average LR of -7.67%. In contrast, RRD improves accuracy over DI while achieving stronger final-output length reduction than all baselines, suggesting that regulating reflection events better balances useful reasoning continuation and redundant reflection suppression.

The advantage of RRD is consistent across mod-

Method	AIME24			AIME25			MATH500			GPQA-D			AVG	
	Acc↑	Len↓	LR↑	Acc↑	Len↓	LR↑	Acc↑	Len↓	LR↑	Acc↑	Len↓	LR↑	Acc↑	LR↑
Qwen3-1.7B														
DI	43.3	16122	—	43.3	15759	—	90.3	5758	—	41.9	9265	—	54.7	—
DEER	39.3	14309	11.2	33.3	9706	38.4	89.7	3588	37.7	38.1	8194	11.6	50.1	24.7
CGRS	44.7	12687	21.3	39.3	9974	36.7	88.9	4386	23.8	38.5	8544	7.8	52.9	22.4
MTI	46.7	15602	3.2	40.7	16786	-6.5	90.5	5706	0.9	39.6	9242	0.2	<u>54.4</u>	-0.5
RRD	48.0	12417	23.0	41.3	10399	34.0	90.6	4557	20.9	39.7	8013	13.5	54.9	<u>22.8</u>
Qwen3-8B														
DI	73.3	13426	—	59.3	15862	—	95.1	5256	—	57.4	9203	—	71.3	—
DEER	66.7	9503	29.2	54.0	11914	24.9	92.7	3424	34.9	59.3	8275	10.1	68.2	24.8
CGRS	68.0	9793	27.1	57.3	13067	17.6	93.5	4007	23.8	59.8	7732	16.0	69.7	21.1
MTI	76.3	20751	-54.6	61.3	15834	0.2	95.4	5419	-3.1	61.5	9169	0.4	73.5	-14.3
RRD	75.7	9387	30.1	60.6	12742	19.7	95.9	3520	33.0	61.1	7984	13.2	<u>73.2</u>	<u>24.0</u>
Qwen3-14B														
DI	70.7	12593	—	64.7	15311	—	96.5	4679	—	63.5	7745	—	73.8	—
DEER	64.7	8642	31.4	61.3	9912	35.3	95.6	3162	32.4	61.3	7244	6.5	70.7	<u>26.4</u>
CGRS	70.0	9060	28.1	62.7	9743	36.4	96.3	3436	26.6	60.3	6886	11.1	72.3	25.5
MTI	72.0	13538	-7.5	68.7	14427	5.8	96.9	4849	-3.6	63.8	7578	2.2	<u>75.3</u>	-0.8
RRD	74.7	9319	26.0	66.7	9684	36.8	96.7	3207	31.4	64.5	6545	15.5	75.6	27.4
DeepSeek-R1-Distill-Qwen-7B														
DI	50.7	12319	—	36.0	14095	—	93.6	4106	—	36.9	8567	—	54.3	—
DEER	43.3	10160	17.5	33.3	11529	18.2	89.5	2412	41.3	26.8	7633	10.9	48.2	22.0
CGRS	49.3	11384	7.6	33.3	11208	20.5	87.9	2407	41.4	32.8	6959	18.8	50.9	<u>22.1</u>
MTI	52.7	13989	-13.6	38.7	12777	9.3	93.3	4414	-7.5	37.4	10177	-18.8	55.5	-7.6
RRD	51.3	8576	30.4	36.0	9548	32.3	92.8	3278	20.2	39.6	6633	22.6	<u>54.9</u>	26.4
DeepSeek-R1-Distill-Llama-8B														
DI	40.7	13564	—	30.0	12500	—	87.7	4629	—	45.3	8405	—	50.9	—
DEER	39.3	9705	28.4	31.3	9386	24.9	82.8	3232	30.2	38.5	7669	8.8	48.0	<u>23.1</u>
CGRS	41.3	10897	19.7	27.3	10015	19.9	85.7	3706	19.9	45.1	7174	14.6	49.9	18.5
MTI	43.3	14041	-3.5	28.0	15939	-27.5	87.5	5434	-17.4	45.0	9413	-12.0	<u>50.9</u>	-15.1
RRD	42.7	9228	32.0	29.3	9720	22.2	86.9	3422	26.1	48.0	6333	24.7	51.7	26.2

Table 1: Main accuracy–output-length results across five LRLMs and four reasoning benchmarks. Acc denotes final-answer accuracy, Len denotes the average number of tokens retained in the final reasoning output, excluding discarded auxiliary probe tokens, and LR denotes final-output length reduction relative to Direct Inference. For AVG, we report average Acc and LR across benchmarks.

els and tasks. Across all five backbones, RRD improves average accuracy over DI while achieving 22.84%–27.42% LR, whereas the baselines show less stable transfer. For example, DEER and CGRS obtain reasonable reductions in final-output length on some Qwen3 models but suffer larger accuracy drops on the DeepSeek-R1-Distill series, while MTI improves accuracy on several settings but consistently increases final-output length. Across benchmarks, RRD improves accuracy on AIME24 and GPQA-Diamond, maintains comparable accuracy on the harder AIME25, and preserves the already high accuracy on MATH500, with consistent final-output length reduction on all four tasks. These results show that RRD is not specialized to

a particular backbone or benchmark, but provides a robust accuracy–output-length trade-off across model families and reasoning domains.

In summary, RRD consistently improves the accuracy–output-length frontier across models, benchmarks, and difficulty levels.

4.3 End-to-End Inference Cost

Table 2 reports the end-to-end inference costs of DI and RRD on MATH500 using Qwen3-8B, DeepSeek-R1-Distill-Qwen-7B, and DeepSeek-R1-Distill-Llama-8B. The corresponding results on AIME24, AIME25, and GPQA-Diamond are provided in Appendix E. For each model, we report final-output tokens, discarded TDS-probe tokens,

Method	Final output (tok./ex.)	Discarded TDS probe (tok./ex.)	RVS answer- probe (tok./ex.)	Total accounted (tok./ex.)	Wall-clock latency (s/ex.)
Qwen3-8B					
DI	5256	—	—	5256	73.16
RRD	3520	136	41	3697	54.83
DeepSeek-R1-Distill-Qwen-7B					
DI	4106	—	—	4106	71.14
RRD	3278	109	36	3423	60.20
DeepSeek-R1-Distill-Llama-8B					
DI	4629	—	—	4629	76.98
RRD	3422	115	40	3577	62.52

Table 2: End-to-end inference cost of DI and RRD on MATH500 across three representative LRLMs. For DI, total accounted tokens equal final-output tokens. For RRD, total accounted tokens include final-output tokens, discarded TDS-probe tokens, and RVS answer-probe tokens. Accepted TDS probes are retained in the final output and are not counted twice. Wall-clock latency includes all auxiliary computation introduced by RRD.

RVS answer-probe tokens, total accounted tokens, and wall-clock latency.

Total accounted tokens are computed as the sum of final-output tokens, discarded TDS-probe tokens, and RVS answer-probe tokens. Accepted TDS probes are retained in the final output and are therefore not counted twice. Wall-clock latency includes the main decoding process, probe generation, continuation-validator inference, and Prefix-Consistency Reliability computation.

Averaged across the three models on MATH500, RRD reduces total accounted tokens by 23.01% and wall-clock latency by 19.74% relative to DI. These results show that RRD retains actual inference savings after accounting for its additional decoding-time computation.

4.4 Ablation Study

Following the two-mechanism design of RRD, we conduct ablation experiments by removing Trajectory Drift Suppression (TDS) and Redundant Verification Suppression (RVS) separately. The *w/o TDS* variant disables trajectory-drift suppression at reflection events, while the *w/o RVS* variant disables reliability-based suppression of answer-adjacent redundant verification. We evaluate these variants on Qwen3-8B using MATH500 and AIME25, and report answer accuracy (Acc), average generation length (Len), and length reduction ratio (LR).

As shown in Table 3, removing either mechanism weakens the overall accuracy–efficiency trade-off. Both ablated variants still outperform Vanilla, indicating that each mechanism independently contributes to more efficient reasoning. However, their effects are different. The *w/o TDS* variant still reduces generation length, but it loses

Method	MATH500			AIME25		
	Acc (%) ↑	Len ↓	LR ↑	Acc (%) ↑	Len ↓	LR ↑
Vanilla	95.1	5256	—	59.3	15862	—
w/o TDS	95.5	4203	20.1	59.4	14023	11.6
w/o RVS	95.8	4479	14.8	59.7	14244	10.2
RRD	95.9	3520	32.8	60.6	12742	19.7

Table 3: Ablation on the two mechanisms of RRD using Qwen3-8B. TDS denotes Trajectory Drift Suppression, RVS denotes Redundant Verification Suppression, and LR is computed relative to Vanilla.

Reliability signal	Acc (%) ↑	Len ↓
Whole-answer confidence C_1	92.7	3946
Entropy-based certainty C_2	91.5	4019
Prefix-Consistency Reliability q_t	95.9	3520

Table 4: Analysis of reliability signals in RVS on Qwen3-8B with MATH500. C_1 denotes whole-answer confidence, C_2 denotes entropy-based certainty, and q_t denotes Prefix-Consistency Reliability.

part of the trajectory-level control provided by TDS, making reflection-induced trajectory shifts less explicitly regulated. The *w/o RVS* variant preserves strong answer accuracy, but produces longer outputs than the full model, suggesting that answer-adjacent verification is not sufficiently suppressed without RVS. In contrast, the full RRD consistently achieves the best overall results on both benchmarks, showing that TDS and RVS address complementary forms of unregulated reflection.

4.5 Analysis

Analysis of Prefix-Consistency Reliability. We compare Prefix-Consistency Reliability q_t with two alternative reliability signals used in RVS. The first

is whole-answer confidence (Yang et al., 2025b):

$$C_1 = \left(\prod_{i=1}^n \max_{v \in V} p_\theta(v \mid x, o_{\leq t}, \mathcal{I}, a_t^{(<i)}) \right)^{1/n}. \quad (9)$$

The second is entropy-based certainty (Huang et al., 2026):

$$C_2 = 1 - \frac{1}{n} \sum_{i=1}^n \frac{H(p_i)}{\log |V|}, \quad (10)$$

where $p_i = p_\theta(\cdot \mid x, o_{\leq t}, \mathcal{I}, a_t^{(<i)})$.

Whole-answer confidence can smooth over unreliable early prefixes, as later high-confidence tokens may compensate for weak initial support. Entropy-based certainty captures distributional sharpness, but low entropy does not necessarily indicate that the answer is supported by the current reasoning prefix. In contrast, q_t evaluates prefix-level reliability and suppresses answer-adjacent reflection only when the probed answer remains reliable throughout generation. This matches the goal of RVS: reducing redundant verification without overly suppressing useful checking. As shown in Table 4, q_t provides the strongest accuracy–efficiency trade-off among the compared signals.

Sensitivity of the answer reliability threshold.

Finally, we analyze the effect of the answer reliability threshold τ_a in the answer-aware reflection suppression stage. Recall that answer-adjacent reflection cues are suppressed only when the prefix-consistency reliability satisfies $q_t \geq \tau_a$. We vary τ_a and report its impact on accuracy and generation length on MATH500 and AIME25.

Figure 3 shows that increasing τ_a makes RVS more conservative, leading to higher accuracy but longer generations on both benchmarks. The trend is milder on MATH500, where accuracy improves early and then becomes relatively stable while length continues to grow. In contrast, AIME25 remains more sensitive to larger thresholds, suggesting that harder problems require stronger reliability before suppressing answer-adjacent reflection. Overall, τ_a controls the trade-off between aggressive compression and preserving useful verification.

4.6 Case Study

Figure 4 shows a representative MATH-500 problem solved by Qwen3-8B. The solution rewrites $r_i \bar{r}_i = |r_i|^2$, parametrizes roots with tenth roots

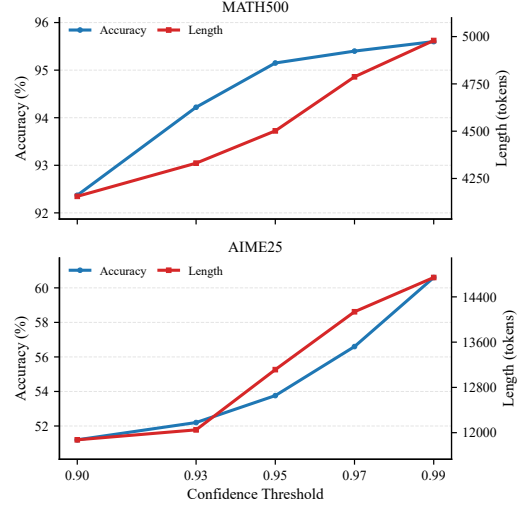


Figure 3: Effect of the reliability threshold τ_a on accuracy and generation length in RVS.

of -1 , and uses conjugate pairing to compute the final answer 850.

In Vanilla, unregulated reflection triggers cause two inefficiencies: drift from the modulus-based trajectory into alternative branches, and verification saturation after a provisional answer, producing redundant checks without new reasoning.

RRD manages reflection events: TDS keeps the reasoning on the modulus-based trajectory, while RVS suppresses redundant verification once the provisional answer is reliable. As a result, the model reaches 850 efficiently along a coherent reasoning trajectory.

5 Conclusion

In this paper, we identify unregulated reflection as a source of inefficient reasoning in LRLMs and characterize two failure modes: Trajectory Drift and Verification Saturation. We propose RRD, an inference-time framework that requires no fine-tuning or parameter updates to the target LRLM that regulates reflection events through Trajectory Drift Suppression and Redundant Verification Suppression. By guiding intermediate reasoning toward validated continuation branches and suppressing redundant answer-adjacent verification, RRD reduces unnecessary reasoning while preserving useful self-correction. Experiments on multiple reasoning benchmarks and LRLMs show that RRD improves the accuracy–efficiency trade-off over direct inference and representative baselines. Our findings highlight the importance of regulating reflection, rather than simply extending reasoning,

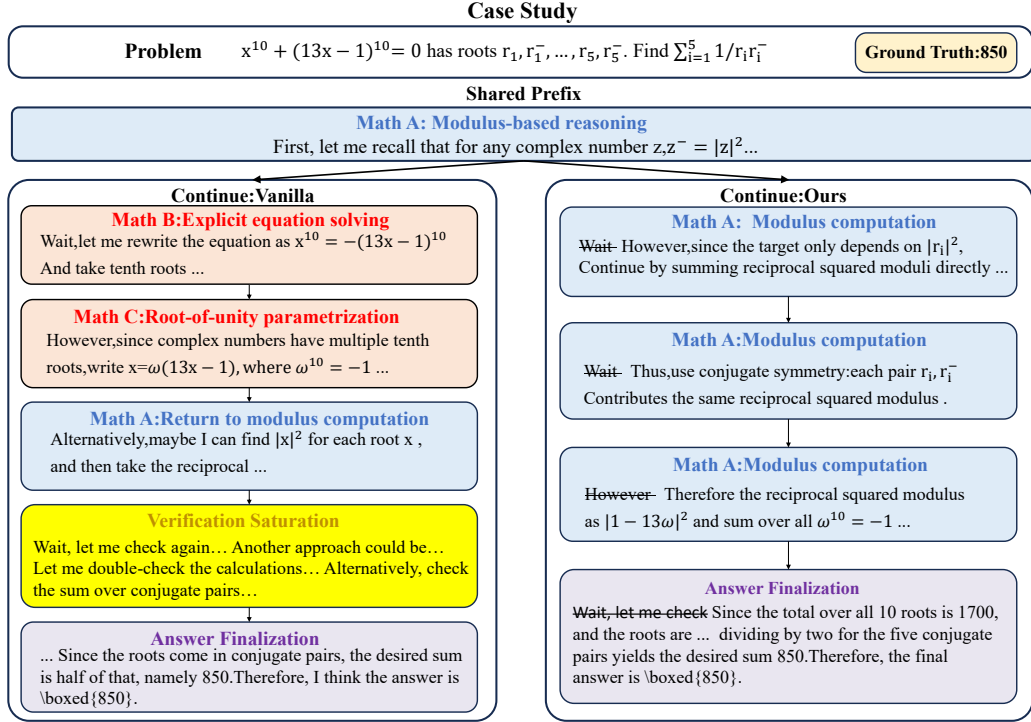


Figure 4: Case study on a MATH-500 problem: Vanilla exhibits trajectory drift and redundant verification, while RRD (TDS + RVS) maintains the reasoning trajectory and efficiently reaches the correct answer 850.

for efficient LRLM inference.

Limitations

While RRD improves the accuracy–efficiency trade-off for LRLMs, the current work still has limitations that warrant further exploration. First, our evaluation requires access to raw reasoning traces and controllable decoding, which restricts the study to open-source models. Since many closed-source reasoning models do not expose their original chain-of-thought traces, it remains unclear whether the same reflection patterns and RRD benefits transfer to such systems. Second, RRD relies on explicit reflection triggers as observable control points. This enables lightweight online regulation, but may miss implicit reflection behaviors that are not marked by surface trigger tokens. Third, RVS uses rule-based provisional answer detection, whose coverage is tied to model-specific answer expressions and output formats. Future work can explore more general reflection detection and answer-state estimation methods to improve robustness across model families. A potential risk is that suppressing reflection may occasionally remove useful self-verification, especially for difficult or high-stakes problems; since RRD is also

coupled with model-specific generation patterns, applying it to sensitive domains or unseen model families requires additional validation.

References

- Pranjal Aggarwal and Sean Welleck. 2025. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*.
- Art of Problem Solving. 2024. 2024 AIME I Problems. https://artofproblemsolving.com/wiki/index.php/2024_AIME_I_Problems. Accessed: 2026-05-19.
- Art of Problem Solving. 2025. 2025 AIME I Problems. https://artofproblemsolving.com/wiki/index.php/2025_AIME_I_Problems. Accessed: 2026-05-19.
- 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, Qiuqi Liu, Mengfei Zhou, Zhuosheng Zhang, and 1 others. 2025.

- Do not think that much for $2+3=?$ on the overthinking of long reasoning models. In *Forty-second International Conference on Machine Learning*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, and 1 others. 2025. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks. *arXiv preprint arXiv:2502.08235*.
- Bowen Ding, Yuhao Chen, Futing Wang, Lingfeng Ming, and Tao Lin. 2025. Do thinking tokens help or trap? towards more efficient large reasoning model. *arXiv preprint arXiv:2506.23840*.
- Chenrui Fan, Ming Li, Lichao Sun, and Tianyi Zhou. 2025. Missing premise exacerbates overthinking: Are reasoning models losing critical thinking skill? *arXiv preprint arXiv:2504.06514*.
- Sicheng Feng, Gongfan Fang, Xinyin Ma, and Xinchao Wang. 2025. Efficient reasoning models: A survey. *arXiv preprint arXiv:2504.10903*.
- Yichao Fu, Junda Chen, Siqi Zhu, Zheyu Fu, Zhongdongming Dai, Aurick Qiao, and Hao Zhang. 2024. Efficiently serving llm reasoning programs with certindex. *arXiv e-prints*, pages arXiv–2412.
- Yichao Fu, Xuwei Wang, Yuandong Tian, and Jiawei Zhao. 2025. Deep think with confidence. *arXiv preprint arXiv:2508.15260*.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2025. Token-budget-aware llm reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24842–24855.
- Jiameng Huang, Baijiong Lin, Guhao Feng, Jierun Chen, Di He, and Lu Hou. 2026. Efficient reasoning for large reasoning language models via certainty-guided reflection suppression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 31176–31184.
- Guochao Jiang, Guofeng Quan, Zepeng Ding, Ziqin Luo, Dixuan Wang, and Zheng Hu. 2025. Flashthink: An early exit method for efficient reasoning. *arXiv preprint arXiv:2505.13949*.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626.
- Yassir Laaouach. 2025. Halt-cot: Model-agnostic early stopping for chain-of-thought reasoning via answer entropy. In *4th Muslims in ML Workshop co-located with ICML 2025*.
- Junyan Li, Wenshuo Zhao, Yang Zhang, and Chuang Gan. 2025a. Steering llm thinking with budget guidance. *arXiv preprint arXiv:2506.13752*.
- Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. 2024. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. In *The Twelfth International Conference on Learning Representations*.
- Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, and 1 others. 2025b. From system 1 to system 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419*.
- Guosheng Liang, Longguang Zhong, Ziyi Yang, and Xiaojun Quan. 2025. **Thinkswitcher: When to think hard, when to think fast**. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 5185–5201. Association for Computational Linguistics.
- Baohao Liao, Yuhui Xu, Hanze Dong, Junnan Li, Christof Monz, Silvio Savarese, Doyen Sahoo, and Caiming Xiong. 2025. Reward-guided speculative decoding for efficient llm reasoning. *arXiv preprint arXiv:2501.19324*.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let’s verify step by step. In *International Conference on Learning Representations*, volume 2024, pages 39578–39601.
- Weizhe Lin, Xing Li, Zhiyuan Yang, Xiaojin Fu, Hui-Ling Zhen, Yaoyuan Wang, Xianzhi Yu, Wulong Liu, Xiaosong Li, and Mingxuan Yuan. 2025. Trimr: Verifier-based training-free thinking compression for efficient test-time scaling. *arXiv preprint arXiv:2505.17155*.
- Xin Liu and Lu Wang. 2025. Answer convergence as a signal for early stopping in reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 17907–17918.
- Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. 2025a. Reasoning models can be effective without thinking. *arXiv preprint arXiv:2504.09858*.

- Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025b. Cot-valve: Length-compressible chain-of-thought tuning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6025–6035.
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. 2025. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20286–20332.
- Tergel Munkhbat, Namgyu Ho, Seo Hyun Kim, Yongjin Yang, Yujin Kim, and Se-Young Yun. 2025. Self-training elicits concise reasoning in large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 25127–25152.
- Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Waleed Kadous, and Ion Stoica. 2024. Routellm: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665*.
- OpenAI. 2024a. Introducing OpenAI o1. <https://openai.com/o1/>. Accessed: 2026-05-18.
- OpenAI. 2024b. Learning to reason with LLMs. <https://openai.com/index/learning-to-reason-with-llms/>. Accessed: 2026-05-18.
- Qianjun Pan, Wenkai Ji, Yuyang Ding, Junsong Li, Shilian Chen, Junyi Wang, Jie Zhou, Qin Chen, Min Zhang, Yulan Wu, and 1 others. 2025. A survey of slow thinking-based reasoning llms using reinforced learning and inference-time scaling law. *arXiv preprint arXiv:2505.02665*.
- Ziqing Qiao, Yongheng Deng, Jiali Zeng, Dong Wang, Lai Wei, Guanbo Wang, Fandong Meng, Jie Zhou, Ju Ren, and Yaoyue Zhang. 2025. Concise: Confidence-guided compression in step-by-step efficient reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 8021–8040.
- Xiaoye Qu, Yafu Li, Zhao-Chen Su, Weigao Sun, Jianhao Yan, Dongrui Liu, Ganqu Cui, Daizong Liu, Shuxian Liang, Junxian He, and 1 others. 2025. A survey of efficient reasoning for large reasoning models: Language, multimodality, and beyond. *arXiv preprint arXiv:2503.21614*.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*.
- Shubhankar Singh. 2024. Jigsaw pieces of meaning: Modeling discourse coherence with informed negative sample synthesis. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 1895–1908.
- 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 arXiv:2408.03314*.
- Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, and 1 others. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*.
- Guoheng Sun, Ziyao Wang, Xuandong Zhao, Bowei Tian, Zheyu Shen, Yexiao He, Jinming Xing, and Ang Li. 2025a. Invisible tokens, visible bills: The urgent need to audit hidden operations in opaque llm services. *arXiv preprint arXiv:2505.18471*.
- Renliang Sun, Wei Cheng, Dawei Li, Haifeng Chen, and Wei Wang. 2025b. Stop when enough: Adaptive early-stopping for chain-of-thought reasoning. *arXiv preprint arXiv:2510.10103*.
- Chenlong Wang, Yuaning Feng, Dongping Chen, Zhaoyang Chu, Ranjay Krishna, and Tianyi Zhou. 2025a. Wait, we don’t need to “wait”! removing thinking tokens improves reasoning efficiency. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 7459–7482. Association for Computational Linguistics.
- Rui Wang, Hongru Wang, Boyang Xue, Jianhui Pang, Shudong Liu, Yi Chen, Jiahao Qiu, Derek Fai Wong, Heng Ji, and Kam-Fai Wong. 2025b. Harnessing the reasoning economy: A survey of efficient reasoning for large language models. *arXiv preprint arXiv:2503.24377*.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.
- Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. 2025. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Qiaowei Li, Minghui Chen, Zheng Lin, and Weiping Wang. 2025b. Dynamic early exit in reasoning models. *arXiv preprint arXiv:2504.15895*.

Zhen Yang, Mingyang Zhang, Feng Chen, Ganggui Ding, Liang Hou, Xin Tao, and Ying-Cong Chen. 2025c. Less is more: Improving llm reasoning with minimal test-time intervention. *arXiv preprint arXiv:2510.13940*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Junyu Zhang, Runpei Dong, Han Wang, Xuying Ning, Haoran Geng, Peihao Li, Xialin He, Yutong Bai, Jitendra Malik, Saurabh Gupta, and 1 others. 2025. Alphaone: Reasoning models thinking slow and fast at test time. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 11340–11365.

Jason Zhu and Hongyu Li. 2025. Towards concise and adaptive thinking in large reasoning models: A survey. *arXiv preprint arXiv:2507.09662*.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, and 1 others. 2025. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. In *International Conference on Learning Representations*, volume 2025, pages 66602–66656.

A Code Availability

We provide an anonymous GitHub repository at: <https://github.com/mqzml/RRD.git>. The repository contains the implementation of RRD, including TDS and RVS, together with the scripts and configurations for reproducing the experiments.

B Data Safety.

We use publicly available mathematical and scientific reasoning benchmarks and model-generated reasoning traces, and do not collect user or human-subject data. We checked the data for personally identifying information and offensive content and found none.

C Reflection Trigger Set

RRD uses explicit reflection triggers as online control points during decoding. These triggers are used to identify reflection events, rather than to segment the full reasoning trace offline. We start from a small set of base trigger keywords and expand each

keyword by matching its natural-language variants in the tokenizer vocabulary. These variants may differ in capitalization, punctuation, leading spaces, or subword forms, depending on the tokenizer. After expansion, we filter out variants that are unlikely to function as reflection triggers in actual reasoning traces. Table 5 lists the base trigger keywords used in our implementation.

This trigger set is not intended to provide an exhaustive definition of reflection. We adopt explicit surface triggers because they can be detected online at low cost during decoding. Consequently, implicit reflection behaviors without explicit surface cues may not be captured by the current trigger-based design.

Reflection Trigger Set
“wait”, “alternatively”, “but”, “check”, “verify”

Table 5: Reflection trigger keywords used to identify reflection events in RRD.

D Continuation Validator for Trajectory Drift Suppression

D.1 Task Definition

The continuation validator is used in Trajectory Drift Suppression (TDS) to determine whether a probe continuation obtained after temporarily masking reflection triggers can extend the original reasoning path. Given a current local reasoning context c and a probe continuation u , the validator predicts a binary label:

$$D(c, u) \rightarrow \{\text{CONTINUE}, \text{REFLECT}\}. \quad (11)$$

Here, c denotes a fixed-length local context before the reflection event. The probe continuation u denotes the bounded continuation generated after suppressing reflection triggers during TDS.

D.2 Dataset Construction

We construct supervision data for the continuation validator from LRLM reasoning trajectories. The source problems are mainly selected from DeepScaleR-derived mathematical reasoning problems, with a small set of non-overlapping scientific and general reasoning problems included for domain robustness. None of the problems used for validator construction overlap with the evaluation benchmarks, including AIME24, AIME25, MATH500, and GPQA-Diamond.

We generate reasoning trajectories using two open-source LRLMs, Qwen3-8B and DeepSeek-R1-Distill-Llama-8B, under direct inference. The generated traces are used only for offline validator supervision. The trained validator is fixed and applied unchanged to all evaluated LRLMs.

For each generated trajectory, we locate reflection events using the trigger set $\mathcal{R}$ and extract a fixed-length local context c before each trigger. We then pair c with a probe continuation u generated after temporarily suppressing the reflection trigger. GPT-5 is used only as an offline annotator to assign binary labels. A context–continuation pair is labeled as CONTINUE if u can serve as a valid next step of the original reasoning path. Otherwise, it is labeled as REFLECT, indicating that the probe continuation should be rejected and the original reflection path should be preserved.

Following local coherence modeling (Singh, 2024), we also construct hard negative continuations from valid context–continuation pairs. We use three negative types: repetitive stalling, implicit reflection, and auxiliary branches. They correspond to continuations that repeat previous reasoning without progress, enter reflection without explicit triggers, or open a side reasoning path that cannot serve as the direct next step. These negative continuations are labeled as REFLECT.

For evaluation, we construct a separate held-out set of 3,000 TDS decision samples generated by the target LRLM families, with 1,000 samples each from the Qwen3, R1-Distill-Qwen, and R1-Distill-Llama series. These samples are held out from validator training and follow the same context–continuation construction and offline annotation protocol described above. The evaluation set is approximately balanced overall, containing 50.9% CONTINUE samples and 49.1% REFLECT samples. Specifically, the respective CONTINUE/REFLECT proportions are 52.1%/47.9% for the Qwen3 series, 49.6%/50.4% for the R1-Distill-Qwen series, and 51.0%/49.0% for the R1-Distill-Llama series.

D.3 Validator Training

We instantiate the continuation validator with DeBERTa-v3-base. Each training instance consists of a local reasoning context c , a probe continuation u , and a binary label $y \in \{\text{CONTINUE}, \text{REFLECT}\}$. We concatenate c and u as a standard paired input and feed it into the validator. The validator outputs

Trace source	#Samples	CONTINUE	REFLECT	Acc.	Macro-F1
Qwen3 series	1,000	52.1%	47.9%	92.5	91.8
R1-Distill-Qwen series	1,000	49.6%	50.4%	91.6	90.2
R1-Distill-Llama series	1,000	51.0%	49.0%	91.3	90.4
All target LRLMs	3,000	50.9%	49.1%	91.8	90.8

Table 6: Class distribution and classification performance of the continuation validator on held-out TDS decision samples. Accuracy and Macro-F1 are reported as percentages.

a probability distribution over the two labels:

$$p_\phi(y \mid c, u) = D_\phi(c, u), \quad (12)$$

where ϕ denotes the parameters of the validator.

We train the validator with the standard cross-entropy loss:

$$\mathcal{L}_{\text{val}} = -\frac{1}{N} \sum_{i=1}^N \log p_\phi(y_i \mid c_i, u_i), \quad (13)$$

where N is the number of training instances. During inference, the validator predicts the label with the highest probability:

$$D(c, u) = \arg \max_y p_\phi(y \mid c, u). \quad (14)$$

D.4 Validator Evaluation

We further examine the two types of validator errors. A false acceptance occurs when a probe labeled as REFLECT is predicted as CONTINUE, causing TDS to suppress a potentially useful reflection. A false rejection occurs when a probe labeled as CONTINUE is predicted as REFLECT, causing TDS to preserve an unnecessary reflection path. Representative cases are shown below.

False acceptance

Local context: ... Because the triangles are equilateral, these three sides have equal length. The perimeter of each triangle is 21 inches, so each side is 7 inches. Therefore, the boundary side of the hexagon is also 7 inches. This relation appears to follow directly from the standard decomposition of a regular hexagon.

Masked trigger: *Wait*

Probe continuation: However, the radii may differ from the boundary side, so the central-angle...

Gold / Prediction: REFLECT / CONTINUE

Impact: TDS accepts a probe that enters an implicit reflection branch instead of directly extending the current reasoning path.

Method	Final output (tok./ex.)	Discarded TDS probe (tok./ex.)	RVS answer- probe (tok./ex.)	Total accounted (tok./ex.)	Wall-clock latency (s/ex.)
Qwen3-8B					
DI	13426	—	—	13426	227.74
RRD	9387	263	61	9711	174.39
DeepSeek-R1-Distill-Qwen-7B					
DI	12319	—	—	12319	209.82
RRD	8576	229	57	8862	158.49
DeepSeek-R1-Distill-Llama-8B					
DI	13564	—	—	13564	235.31
RRD	9228	234	55	9517	163.68

Table 7: End-to-end inference cost of DI and RRD on AIME24 across three representative LRLMs.

Method	Final output (tok./ex.)	Discarded TDS probe (tok./ex.)	RVS answer- probe (tok./ex.)	Total accounted (tok./ex.)	Wall-clock latency (s/ex.)
Qwen3-8B					
DI	15862	—	—	15862	288.16
RRD	12742	219	62	13023	234.45
DeepSeek-R1-Distill-Qwen-7B					
DI	14095	—	—	14095	260.97
RRD	9548	241	53	9842	184.54
DeepSeek-R1-Distill-Llama-8B					
DI	12500	—	—	12500	249.03
RRD	9720	198	62	9980	201.60

Table 8: End-to-end inference cost of DI and RRD on AIME25 across three representative LRLMs.

False rejection

Local context: ...The remaining two angles are also 60° , so the triangle is equilateral. Hence, each radius equals the corresponding boundary side of the hexagon. Since the equilateral triangle has perimeter 21 inches, each of these equal sides has length 7 inches.

Masked trigger: *Wait*

Probe continuation: Thus, the corresponding side of the regular hexagon also has length 7 inches, allowing the perimeter calculation...

Gold / Prediction: CONTINUE / REFLECT

Impact: TDS rejects a valid continuation and restores the original reflection path, reducing the resulting token savings.

False acceptances cause TDS to retain invalid probe continuations, potentially disrupting the reasoning trajectory, whereas false rejections restore the original reflection path despite a valid probe, reducing token savings.

Table 6 reports the class distribution and classification performance of the continuation validator on the held-out evaluation set.

The validator achieves over 91% accuracy and over 90% Macro-F1 across all three target LRLM families. The small variation across trace sources indicates stable cross-model classification performance.

E Full End-to-End Inference Cost Results

We complement the MATH500 results in Table 2 with end-to-end inference-cost results on AIME24, AIME25, and GPQA-Diamond. For each benchmark, we compare DI and RRD using Qwen3-8B, DeepSeek-R1-Distill-Qwen-7B, and DeepSeek-R1-Distill-Llama-8B. We report final-output tokens, discarded TDS-probe tokens, RVS answer-probe tokens, total accounted tokens, and wall-clock latency.

The accounting follows the same definition as in the main text. For DI, total accounted tokens equal final-output tokens. For RRD, total accounted tokens include final-output tokens, discarded TDS-probe tokens, and RVS answer-probe tokens. Accepted TDS probes are already included in the final output, while wall-clock latency includes all auxiliary computation introduced by RRD.

AIME24. Table 7 reports the inference-cost results on AIME24. Averaged across the three models, RRD reduces total accounted tokens by 28.52% and wall-clock latency by 26.11% relative to DI.

AIME25. Table 8 reports the corresponding results on AIME25. Averaged across the three models, RRD reduces total accounted tokens by 22.74% and wall-clock latency by 22.32% relative to DI.

Method	Final output (tok./ex.)	Discarded TDS probe (tok./ex.)	RVS answer- probe (tok./ex.)	Total accounted (tok./ex.)	Wall-clock latency (s/ex.)
Qwen3-8B					
DI	9203	—	—	9203	124.81
RRD	7984	115	34	8133	109.58
DeepSeek-R1-Distill-Qwen-7B					
DI	8567	—	—	8567	117.59
RRD	6633	121	37	6781	95.04
DeepSeek-R1-Distill-Llama-8B					
DI	8405	—	—	8405	111.29
RRD	6333	146	48	6527	89.15

Table 9: End-to-end inference cost of DI and RRD on GPQA-Diamond across three representative LRLMs.

GPQA-Diamond. Table 9 reports the corresponding results on GPQA-Diamond. Averaged across the three models, RRD reduces total accounted tokens by 18.27% and wall-clock latency by 17.09% relative to DI.

Overall results. Across the nine model-benchmark combinations reported in this appendix, RRD reduces total accounted tokens by 23.18% and wall-clock latency by 21.84% on average. Together with the MATH500 results in the main text, these results show that RRD maintains actual inference savings across different models and reasoning benchmarks after accounting for its auxiliary decoding-time computation.

F Character-Level Rules for RVS Activation

RVS is applied only after a provisional answer has appeared in the current reasoning prefix. To verify that this condition can be identified with lightweight rules, we analyze model-generated reasoning traces around provisional answers. For each case, we consider a two-sentence local window

$$w_i = (s_i, s_{i+1}), \quad (15)$$

where s_i is the sentence containing the provisional answer and s_{i+1} is the following sentence. We observe that this local region follows a stable surface pattern: the provisional-answer sentence usually contains an answer-introducing expression, while the following sentence often contains a verification expression. This motivates a simple character-level activation rule for RVS.

We derive two sets of substring patterns from these windows. The first set, $\mathcal{P}_{\text{ans}}$, contains answer-introducing patterns that appear in s_i ; the second set, $\mathcal{P}_{\text{ver}}$, contains verification patterns that appear in s_{i+1} . In $\mathcal{P}_{\text{ans}}$, $\langle \text{target} \rangle$ is instantiated from the

Rule set	Character-level templates
$\mathcal{P}_{\text{ans}}$	$\langle \text{target} \rangle \langle \text{link} \rangle$
	$\langle \text{link} \rangle \langle \text{target} \rangle$
	answer $\langle \text{link} \rangle$
	result $\langle \text{link} \rangle$
	total $\langle \text{link} \rangle$
$\mathcal{P}_{\text{ver}}$	check
	double-check
	verify
	confirm
	make sure

Table 10: Character-level rule sets used to identify the provisional-answer condition for RVS.

Diagnostic Quantity	Count per Example
TD events in DI traces	16.1
VS events in DI traces	7.6
TDS interventions in RRD	9.0
RVS interventions in RRD	2.7

Table 11: Average occurrence and intervention frequencies on Qwen3-8B with MATH500. TD and VS denote detected events in DI traces, whereas TDS and RVS denote actual interventions during RRD decoding.

question, e.g., “angle” for an angle-related question and “price” for a price-related question; $\langle \text{link} \rangle$ denotes flexible linking expressions such as “is”, “should be”, “would be”, “equals”, “=”, “:”, and other similar linking expressions. Patterns in $\mathcal{P}_{\text{ver}}$ are matched as substrings. We keep only recurring and model-general patterns, and remove problem-specific fragments such as particular numbers, entities, or accidental long substrings. Table 10 lists the rule sets used in our implementation. We consider the provisional-answer condition to be satisfied only when s_i matches at least one pattern in $\mathcal{P}_{\text{ans}}$ and s_{i+1} matches at least one pattern in $\mathcal{P}_{\text{ver}}$. Since patterns in $\mathcal{P}_{\text{ver}}$ all serve the same role of

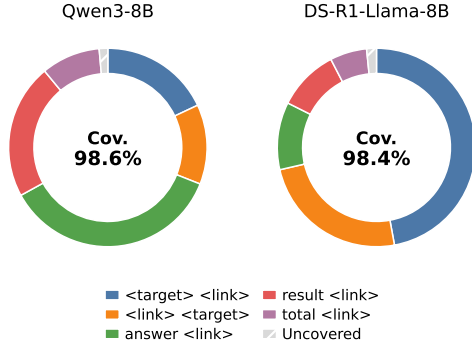


Figure 5: Answer-introducing pattern distributions for identifying the provisional-answer condition. Each donut chart corresponds to one model and reports the distribution of matched patterns in $\mathcal{P}_{\text{ans}}$ among local windows that also satisfy the verification-expression condition. The center label reports the overall coverage of the character-level activation rule.

indicating subsequent verification, we treat them as a unified verification-expression condition in the following distribution analysis.

Figure 5 compares the answer-introducing pattern distributions of Qwen3-8B and DeepSeek-R1-Distill-Llama-8B among local windows satisfying the verification-expression condition. Although the two models prefer different surface forms for expressing provisional answers, the rules in Table 10 cover nearly all analyzed local windows, reaching 98.6% and 98.4% coverage, respectively. This supports the use of compact character-level rules to identify the provisional-answer condition across model families.

G Failure-Mode and Intervention Diagnostics

We analyze the occurrence of trajectory drift (TD) and verification saturation (VS), as well as the intervention frequencies of TDS and RVS, on Qwen3-8B with MATH500. For DI, we report the average number of detected TD and VS events per example. For RRD, we report the average number of actual TDS and RVS interventions per example.

As shown in Table 11, TD and VS events occur repeatedly in DI reasoning trajectories rather than appearing only as isolated cases. TDS and RVS also intervene regularly during RRD decoding, indicating that both components are actively used.

We do not assign a separate help-or-hurt label to each intervention. Because autoregressive decoding is path-dependent, an intervention changes the

subsequent reasoning trajectory, making its counterfactual outcome unavailable within the same generation. We therefore evaluate the aggregate effects of TDS and RVS through the full-run results and component ablations reported in the main paper.