Latent Reward Steering: An Adaptive Inference-Time Framework that Implicitly Promotes Cognitive Behaviors in Reasoning LLMs

Jiakang Li[1]* Guanyu Zhu[2]* Can Jin[1]* Chenxi Huang[3]
Dexu Yu[4] Ronghao Chen[5] Yang Zhou[1] Hongwu Peng[6]
Xuanqi Lan[7] Dimitris N. Metaxas[1]† Youhua Li[8]†

[1]Rutgers University [2]South China Agricultural University [3]Columbia University
[4]Fenz.AI [5]QuantaAlpha [6]Adobe [7]Santa Clara University
[8]City University of Hong Kong

{jiakang.li@rutgers.edu, youhuali2-c@my.cityu.edu.hk, dnm@cs.rutgers.edu}

Abstract

Strong reasoning depends not only on model knowledge but also on how effectively cognitive behaviors are deployed during generation. Existing methods often rely on explicit behavior-level control, making them insufficiently adaptive when failures and required corrections vary across reasoning states, tasks, and models. To this end, we propose Latent Reward Steering (LRS), an adaptive inference-time framework that promotes cognitive behaviors by optimizing the sparse-autoencoder (SAE) latent states that implicitly carry them. Rather than relying on predefined cognitive behaviors or steering directions derived from them, LRS trains a latent reward model on reasoning traces by final answer correctness to estimate the quality of intermediate latent states. During inference, reward gradients provide state-specific correction directions for fragile latent states, while a reward and confidence gate restricts intervention to states the reward signal flags as fragile. Experiments on multiple reasoning LLM backbones and benchmarks show that LRS consistently improves performance over various baselines, and post-hoc analyses further indicate that LRS implicitly promotes good cognitive behaviors that fix the original reasoning errors. Codes are available at: https://github.com/jiakanglee/Latent-Reward-Steering.

1 Introduction

Performing step-by-step reasoning to solve complex problems has become a central research focus in large language models (Wei et al., 2022; Kojima et al., 2022). Yet even strong reasoning models remain brittle: a single early mistake, such as a flawed assumption or a skipped verification step, can gradually derail an otherwise promising reasoning chain (Gan et al., 2025; Huang et al., 2023; Tyen et al., 2024). Recent work highlights cognitive behaviors such as verification, backtracking, and subgoal setting as important ingredients of successful reasoning (Gandhi et al., 2025). This suggests that some reasoning failures are not purely failures of model knowledge, but failures to induce cognitive behaviors at the right moments within an ongoing reasoning chain.

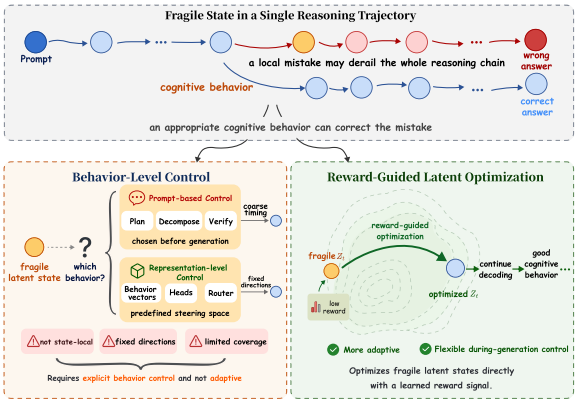

Figure 1: Motivation. A fragile reasoning state can derail reasoning, while explicit behavior-level control may suffer from fixed labels and directions. LRS instead optimizes fragile latent states with a learned reward signal and implicitly promotes useful cognitive behaviors.

The importance of cognitive behaviors in reasoning LLMs has motivated a line of work on controlling such behaviors, most of which involve explicit behavior-level control. Prompt-based methods elicit desired cognitive behaviors through textual instructions, from few-shot in-context learning (Brown et al., 2020) and chain-of-thought (COT) prompting (Wei et al., 2022; Kojima et al., 2022) to more specific behaviors such as sub-goal decomposition (Zhou et al., 2022; Wang et al., 2023), strategic planning (Zheng et al., 2024), and verification

* Equal contribution.
† Equal corresponding authors.

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 13748–13767
October 24-29, 2026 ©2026 Association for Computational Linguistics

(Weng et al., 2023; Miao et al., 2023; Dhuliawala et al., 2024). Representation-level steering methods instead intervene directly on latent states (Turner et al., 2023; Zou et al., 2023), by associating cognitive behaviors with steering directions (Chen et al., 2025), head-specific interventions (Zhang et al., 2025), or routed behavior-vector libraries (Ye et al., 2026).

Compared to prompt-based methods, which explicitly designate particular cognitive behaviors in text before generation begins and cannot target the step where an error actually occurs, representation-level steering methods intervene directly during decoding, offering a more direct and promising space for cognitive behavior control. However, this advantage in interface does not translate into adaptivity: representation-level methods still follow the same explicit behavior-level paradigm as prompt-based ones, where the behaviors to control are predefined and represented as behavior-specific intervention objects such as steering directions, selected heads, or vector libraries. Such a paradigm is not adaptive, since it relies on predefined cognitive behaviors (e.g., verification, backtracking) that may not apply uniformly across models (Gandhi et al., 2025), and on steering directions derived from these predefined behaviors that may not match the local reasoning state (Chen et al., 2025). As a result, although latent-level intervention is a promising direction, both prompt-based and representation-level methods remain insufficiently adaptive when failures and required corrections vary across reasoning tasks and models.

This motivates an interesting research question: *Can we promote good cognitive behaviors adaptively at the latent level without committing to predefined behaviors or their derived steering directions?* Recent findings make this question plausible. First, sparse latent states can be viewed as an internal space where the model's ongoing deployment of cognitive behaviors is implicitly represented (Wang et al., 2026). Second, useful cognitive mechanisms often already exist in the model's latent space, and gains can come from deploying them better rather than from injecting new behaviors (Venhoff et al., 2025a; Ward et al., 2025). Third, recent work shows that latent representations themselves encode reward-like quality signals that can be recovered by a learned model (Du et al., 2025). Building on these observations, we therefore hypothesize that reward-guided optimization of latent states can adaptively promote the good cognitive behaviors already encoded in latent states during reasoning, without ever using explicit behavior control.

Based on this hypothesis, we propose LRS, an adaptive inference-time framework that promotes cognitive behaviors by directly optimizing SAE latent states (Cunningham et al., 2023; Templeton et al., 2024) with a learned reward signal. Rather than relying on predefined cognitive behaviors or steering directions derived from them, LRS trains a latent reward model on successful and unsuccessful reasoning traces to estimate the quality of intermediate latent states. During inference, the reward gradient supplies a state-specific correction direction for the current latent, while a reward and confidence gate restricts intervention to states the reward signal flags as fragile-prone, helping preserve reasoning steps that are already likely to be healthy.

Our main contributions are summarized below:

- To the best of our knowledge, we are the first to frame cognitive behavior control for LLM reasoning as *implicit latent-state optimization*, shifting the focus from explicitly selecting predefined behaviors to adaptively optimizing latent states that represent ongoing cognitive behavior deployment.
- We introduce **LRS**, an adaptive inference-time framework that steers fragile SAE latent states through reward-guided correction together with reward and confidence gating, without relying on predefined cognitive behaviors or their derived steering directions.
- We show that LRS consistently improves inference-time reasoning across multiple LLMs and challenging benchmarks, while qualitative and case-level analyses suggest that it implicitly promotes helpful cognitive behaviors such as solution verification and course correction.

2 Related Work

Inference-time reasoning and prompt-based behavior control. Inference-time reasoning has become an important way to improve LLM performance on complex tasks (Jin et al., 2026; Chen et al., 2026). Few-shot in-context learning and COT prompting elicit general step-by-step reasoning behavior (Brown et al., 2020; Wei et al., 2022; Kojima et al., 2022). Later prompting methods target more specific cognitive behaviors, including

sub-goal decomposition (Zhou et al., 2022; Wang et al., 2023), strategic planning (Zheng et al., 2024), and verification (Weng et al., 2023; Miao et al., 2023; Dhuliawala et al., 2024).

Representation-level steering for reasoning. Activation steering and representation engineering provide a more direct way to influence model behavior by modifying internal states during generation (Turner et al., 2023; Zou et al., 2023). Recent work applies this idea to reasoning control. SEAL decomposes reasoning traces into components such as execution, reflection, and transition, and learns steering vectors to calibrate them (Chen et al., 2025). CREST identifies attention heads associated with behaviors such as verification and backtracking, and derives head-specific steering directions (Zhang et al., 2025). RISER builds a reusable library of reasoning vectors and learns a router to compose them during inference (Ye et al., 2026). These methods make representation-level intervention a promising interface for cognitive behavior control, but remain tied to predefined behaviors, selected heads, fixed directions, or finite vector libraries.

Cognitive behaviors and implicit latent-state optimization. Recent studies highlight the role of cognitive behaviors in LLM reasoning. Cognitive behaviors such as verification, backtracking, and subgoal setting are important ingredients of strong reasoning performance (Gandhi et al., 2025). Other work suggests that useful reasoning mechanisms may already exist in base models, and that gains can come from better deployment of these mechanisms rather than from adding new knowledge (Venhoff et al., 2025a; Ward et al., 2025; Venhoff et al., 2025b). These findings motivate cognitive behavior control, but also reveal the limitation of explicit behavior-level methods: predefined behaviors may not apply uniformly across models, and fixed intervention directions may not match the current reasoning state. In contrast, LRS frames cognitive behavior control as implicit latent-state optimization, using latent states to adaptively steer fragile reasoning states during decoding.

3 Method

3.1 Problem Setup

We formalize the problem of *adaptive cognitive behavior promotion for LLM reasoning* as an inference-time intervention on the model's latent states. Let a parameter-frozen reasoning model $\mathcal{M}$ process an input prompt x and autoregressively generate a response $y = (y_1, \ldots, y_T)$ token by token. At generation step t, let $h_t \in \mathbb{R}^{d_h}$ denote the hidden activation at layer ℓ. As motivated in Section 1, we view h_t as an internal state that implicitly carries the model's intrinsic deployment of cognitive behaviors.

SAE-based intervention space. Directly steering the dense activation h_t is difficult because it is high-dimensional and entangled. We therefore use a pretrained sparse autoencoder (SAE) to map h_t into a low-dimensional sparse latent representation:

$$z_t = f_{\text{enc}}(h_t), \qquad \hat{h}_t = f_{\text{dec}}(z_t), \tag{1}$$

where $z_t \in \mathbb{R}^{d_z}$ is a d_z-dimensional sparse code. We use the pretrained SAEs released by Venhoff et al. (2025a), which were trained on each reasoning model's hidden activations and provide a model-specific sparse code: $d_z = 10$ for Open-Reasoner-7B (Hu et al., 2026) and $d_z = 5$ for Open-Reasoner-1.5B (Hu et al., 2026). Prior work suggests that SAE latents expose interpretable internal features in language models (Cunningham et al., 2023; Templeton et al., 2024), and that SAE latent dimensions can be used to identify and analyze reasoning-relevant cognitive behaviors such as verification, backtracking, and constraint checking in the model's internal states (Venhoff et al., 2025a; Ward et al., 2025; Wang et al., 2026). We therefore use SAE latents as the intervention space for LRS.

Inference-time steering objective. Given a decoding step t, our goal is to obtain a corrected latent state $z'_t = z_t + \Delta z_t$ that improves the likelihood of a correct final answer when decoding continues from the corresponding hidden state. The update is adaptive to the current state, free from predefined cognitive behavior labels or behavior-specific steering directions, and selective enough to avoid disrupting already healthy reasoning states.

3.2 Method Overview

As shown in Figure 2, LRS has three stages. It first constructs SAE latent reasoning traces from a frozen reasoning model and labels each trace by final-answer correctness. It then trains a latent reward model to estimate the quality of intermediate latent states from successful and unsuccessful traces, without using explicit cognitive behavior annotations. During inference, LRS uses the learned

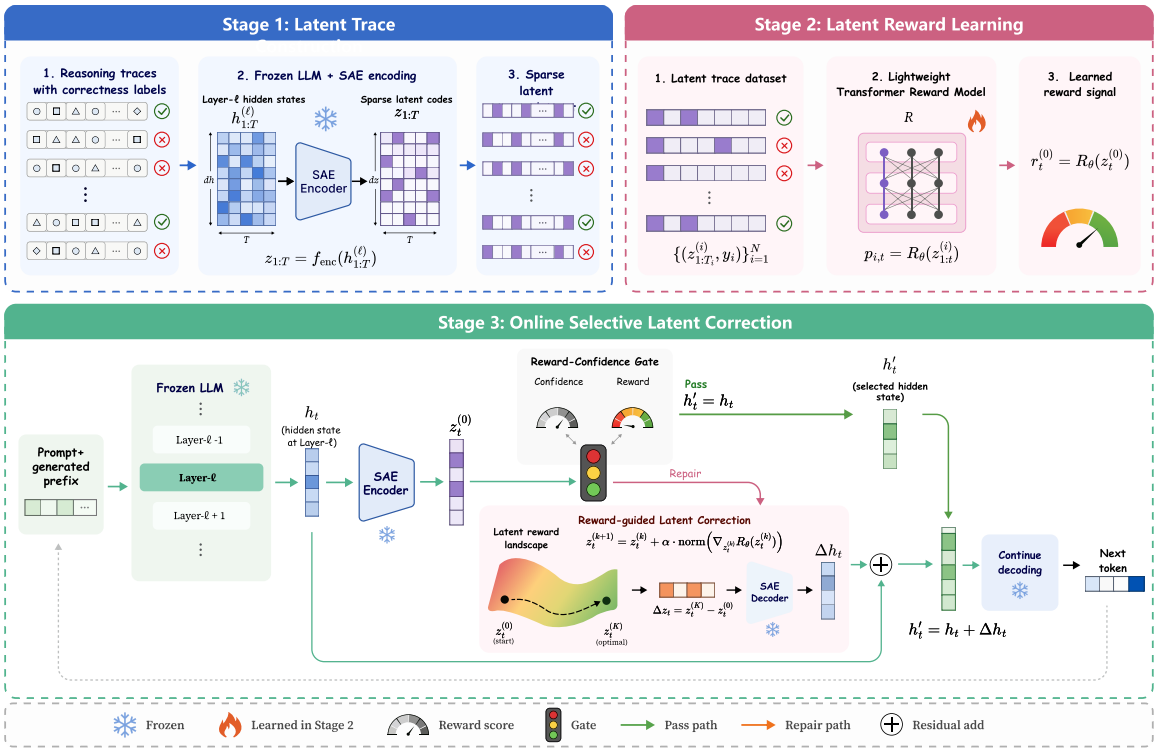

Figure 2: Framework of LRS. It first constructs SAE latent traces from reasoning trajectories, then trains a latent reward model to estimate intermediate-state quality. During inference, a reward and confidence gate identifies fragile states, and reward-guided latent correction updates the hidden state before decoding continues.

reward signal to identify fragile states and applies reward-guided latent correction only when the reward and confidence gate triggers intervention. The decoded latent residual is added back to the hidden activation, enabling adaptive steering of fragile reasoning states without predefined behavior labels or behavior-specific directions.

3.3 Latent Reward Training

We first construct a dataset of sparse latent reasoning traces. For each solved example, we store a latent sequence together with a binary label indicating whether the final answer is correct. Formally, each example is represented as

$$\mathcal{S}_i = (z_{i,s_i}, z_{i,s_i+1}, \ldots, z_{i,T_i}), \qquad y_i \in \{0,1\}, \tag{2}$$

where s_i is the beginning of the reasoning segment.

We train a lightweight Transformer reward model R_θ over these latent sequences:

$$p_{i,t} = R_\theta(z_{i,s_i:t}) \in (0,1), \tag{3}$$

where $p_{i,t}$ is the predicted probability that the reasoning trace belongs to a correct sample. In implementation, the binary label y_i is repeated over all positions of the sequence, and the model is trained using binary cross-entropy:

$$\mathcal{L}_{\text{RM}} = -\sum_i \sum_{t=s_i}^{T_i} \Big[y_i \log p_{i,t} + (1-y_i)\log(1-p_{i,t}) \Big]. \tag{4}$$

To mitigate class imbalance, we apply weighted random sampling during training.

The reward model itself is a small Transformer encoder with LayerNorm on the latent input, a learned embedding layer, positional encoding, two Transformer blocks, and an MLP head.

3.4 Online Reward-Guided Latent Correction

At generation time, we intervene only on generation tokens, not on the prompt prefix. Given the current activation h_t, we encode it into a sparse latent vector built from the SAE encoder:

$$z_t^{(0)} = f_{\text{enc}}(h_t). \tag{5}$$

We then evaluate the reward model on the current latent (implemented as a sequence of length 1 for efficiency) and obtain an initial reward score

$$r_t^{(0)} = R_\theta(z_t^{(0)}). \tag{6}$$

If steering is triggered at this step ($\mathbb{I}_{\text{steer}}(t) = 1$, with the gating rule defined in Section 3.5), we optimize the latent by normalized gradient ascent for K steps:

$$z_t^{(k+1)} = z_t^{(k)} + \alpha \frac{\nabla_{z_t^{(k)}} R_\theta(z_t^{(k)})}{\left\| \nabla_{z_t^{(k)}} R_\theta(z_t^{(k)}) \right\|_2 + \varepsilon}, \quad (7)$$

where α is the step size and K is the number of steering iterations.

Instead of decoding the entire optimized latent, we only decode the latent *difference*

$$\Delta z_t = z_t^{(K)} - z_t^{(0)}, \quad (8)$$

and project it back to activation space with the SAE decoder matrix W_{dec}:

$$\Delta h_t = \Delta z_t W_{\text{dec}}, \qquad h'_t = h_t + \Delta h_t. \quad (9)$$

The steered hidden state h'_t is then fed into the subsequent layers of the LLM.

3.5 Selective Reward and Confidence Gating

Applying latent updates at every generation step may disrupt already-correct reasoning, as the ungated variant LRS BASIC degrades on several benchmarks (Table 1). We therefore introduce a selective gate based on two signals: the current reward score $r_t = R_\theta(z_t^{(0)})$ and the previous-token decoding confidence c_{t-1}, defined as the maximum softmax probability at step $t-1$.

Steering is triggered when either signal suggests a fragile local state:

$$\mathbb{I}_{\text{steer}}(t) = \begin{cases} 1, & \text{if } r_t < \tau_r, \\ 1, & \text{if } r_t \geq \tau_r \text{ and } c_{t-1} < \tau_c, \\ 0, & \text{otherwise}, \end{cases} \quad (10)$$

where τ_r and τ_c are the reward and confidence thresholds. Low reward indicates that the current latent state is unlikely to lead to a correct answer, while low decoding confidence suggests local uncertainty even when the reward is acceptable. Thus, LRS intervenes only when correction is needed and preserves states that appear already on track.

4 Experiments

In this paper, we study the following research questions:

- **RQ1:** Can LRS improve reasoning performance?
- **RQ2:** Does the latent reward model provide a meaningful correction signal?
- **RQ3:** Do LRS-steered traces implicitly promote useful cognitive behaviors?

We further conduct ablation and sensitivity analysis to examine the role of selective intervention and steering strength.

4.1 Experimental Setup

We evaluate LRS with OPEN-REASONER-7B as the primary base model and additionally include OPEN-REASONER-1.5B to assess generalization across model scales. We adopt the pretrained SAE checkpoints released by Venhoff et al. (2025a): steering is applied at layer 20 in a 10-dimensional SAE latent space for OPEN-REASONER-7B, and in a 5-dimensional SAE latent space for OPEN-REASONER-1.5B (Hu et al., 2026). The main results in Table 1 are reported for both backbones, while most qualitative and diagnostic analyses (reward-score separation, post-hoc cognitive behavior annotation, SAE interpretability, efficiency, and case studies) are conducted on the primary OPEN-REASONER-7B model (Hu et al., 2026).

We evaluate on six reasoning benchmarks: MATH-500 (Hendrycks et al., 2021), AIME24 (Art of Problem Solving, 2024a,b), AIME25 (Art of Problem Solving, 2025a,b), AMC23 (math-ai, 2025), INEQMATH (Sheng et al., 2026), and GPQA-DIAMOND (Rein et al., 2023). All benchmarks are evaluated in a zero-shot setting with greedy decoding and batch size 1. More detailed training and experimental settings are provided in Appendix A

4.2 Main Results: LRS Improves Reasoning Performance (RQ1)

LRS improves reasoning across model scales. As shown in Table 1, full LRS improves over standard zero-shot decoding on *all* six reasoning benchmarks for OPEN-REASONER-7B, with gains ranging from +4.4 on the MATH-500 to +14.0 on the INEQMATH, and we observe the same trend on the smaller same family model OPEN-REASONER-1.5B, indicating that the method generalizes across model scales. These improvements all come from inference-time latent steering alone, without any training on the model weights.

Recovery exceeds degradation. To better understand LRS's effect on reasoning improvement at the trace level, we perform a matched-pair analysis

Dataset	OPEN-REASONER-7B					OPEN-REASONER-1.5B				
	Base 0-shot	CoT	5-shot	LRS BASIC 0-shot	LRS 0-shot	Base 0-shot	CoT	5-shot	LRS BASIC 0-shot	LRS 0-shot
MATH-500	79.4	81.4	81.0	83.0 (+3.6)	**83.8** (+4.4)	59.2	58.6	57.2	59.0 (−0.2)	**60.8** (+1.6)
AIME24	16.6	13.3	13.3	16.6 (+0.0)	**26.6** (+10.0)	3.3	6.7	6.7	0.0 (−3.3)	**13.3** (+10.0)
AIME25	16.6	13.3	10.0	20.0 (+3.4)	**26.6** (+10.0)	3.3	0.0	3.3	0.0 (−3.3)	**6.6** (+3.3)
GPQA-DIAMOND	32.3	35.9	38.4	30.8 (−1.5)	**39.4** (+7.1)	18.2	17.2	17.2	15.7 (−2.5)	**22.8** (+4.6)
AMC23	50.0	55.0	**65.0**	45.0 (−5.0)	60.0 (+10.0)	30.0	32.5	30.0	32.5 (+2.5)	**37.5** (+7.5)
INEQMATH	46.0	48.0	48.0	52.0 (+6.0)	**60.0** (+14.0)	29.0	30.0	28.0	28.0 (−1.0)	**34.0** (+5.0)

Table 1: Main results are all reported under greedy decoding with maximum token budget 4000. Values in parentheses denote absolute gains of LRS BASIC / LRS over the corresponding Base model. CoT and few-shot columns report accuracies under chain-of-thought and few-shot prompting, respectively. LRS BASIC is ungated, while full LRS uses reward and confidence gating.

Transition	Base → LRS	Rate
Improved	Wrong → Correct	17.3%
Degraded	Correct → Wrong	7.8%
Preserved	Correct → Correct	28.6%
Unresolved	Wrong → Wrong	46.3%

Table 2: Matched outcome transitions between base and LRS trace ($N = 398$, OPEN-REASONER-7B).

between base and LRS reasoning chains ($N = 398$, OPEN-REASONER-7B). As shown in Table 2, LRS recovers 17.3% of examples from wrong to correct while degrading only 7.8% from correct to wrong, with a net positive effect of $+9.5\%$. This indicates that the gains in Table 1 come from LRS actively recovering failed reasoning traces rather than randomly perturbing them: when LRS intervenes, it is more than twice as likely to fix a wrong reasoning chain as to break a correct one.

LRS consistently outperforms prompt-based behavior control. LRS outperforms both CoT and few-shot prompting on five of the six benchmarks for OPEN-REASONER-7B and on all six for OPEN-REASONER-1.5B. In contrast, prompt-based baselines are unstable: CoT and few-shot prompting underperform standard zero-shot decoding on several benchmarks (e.g., −3.3 for CoT on AIME24 and AIME25 with OPEN-REASONER-7B), consistent with observations that reasoning-tuned models already internalize CoT-style reasoning behavior so adding explicit instructions does not necessarily help (Venhoff et al., 2025a). These results empirically answer RQ1: LRS improves reasoning performance across models and benchmarks.

4.3 Latent Reward Signal Quality (RQ2)

We next examine whether the learned latent reward model captures a signal that is informative of reasoning quality. For each generated reasoning trace, the reward model assigns token-level sigmoid scores over the SAE latent sequence. Although trained only with trace-level correctness, the reward model uses token-level scores because local reasoning errors often propagate, making intermediate latent quality useful for selective correction. We summarize these scores using two diagnostic statistics: the final-token reward p_{last}, which reflects the reward estimate at the end of generation, and the trace-level mean reward p_{mean}, which reflects the average quality signal across the reasoning process.

Figure 3 shows that correct reasoning traces generally receive higher reward scores than incorrect reasoning traces across the evaluated datasets. The separation is most pronounced on AMC23, AIME24, and AIME25, while GPQA-DIAMOND and INEQMATH show the same ordering with smaller gaps. This suggests that the reward model captures reasoning-quality differences in latent space. Importantly, this analysis is not final-answer verification and not behavior classification: the reward model is trained only with final correctness labels over latent traces, without behavior annotations, answer-extraction labels, or predefined cognitive categories. This empirically answers RQ2, showing that the latent reward model provides a meaningful correction signal.

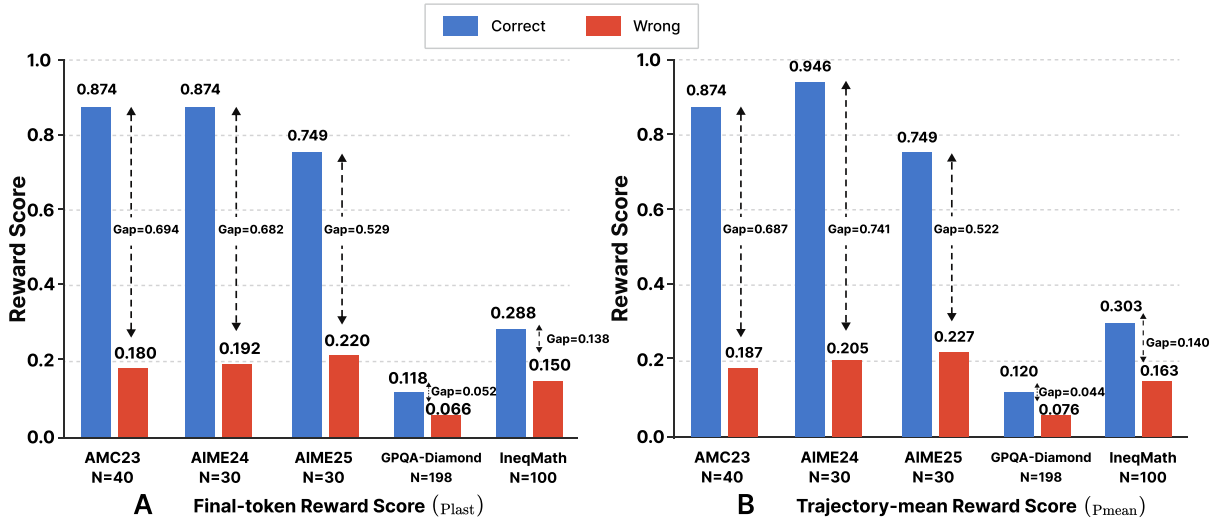

Figure 3: Latent reward scores distinguish successful and failed reasoning traces. We compare final-token reward p_{last} and trace-mean reward p_{mean} between correct and incorrect generations.

4.4 LRS Implicitly Promotes Useful Cognitive Behaviors (RQ3)

We finally examine whether reward-guided latent steering is associated with interpretable changes in cognitive behavior. This analysis is diagnostic only: behavior labels are never used to train the reward model, construct steering vectors, trigger the reward and confidence gate, or guide inference. Instead, we annotate matched base and LRS reasoning chains post hoc to examine whether latent reward steering changes the observable reasoning process by promoting useful cognitive behaviors. We conduct a matched-pair analysis between base and LRS reasoning chains ($N = 398$, OPEN-REASONER-7B) and annotate five cognitive behaviors following Gandhi et al. (2025): *Strategic Planning*, *Structured Decomposition*, *Constraint Grounding*, *Course Correction*, and *Solution Verification*. Annotations are produced by GPT-4o using a fixed post-hoc judging prompt, which is provided in Appendix B. For each trace, a behavior is counted as present if it appears at least once.

Post-hoc behavior frequency. Figure 5 reports the occurrence rates of the five annotated cognitive behaviors in matched base and LRS reasoning chains. LRS-steered traces show higher occurrence of *Course Correction* (+0.20), *Constraint Grounding* (+0.10), and *Solution Verification* (+0.10). These behaviors are closely related to local reasoning correction: revisiting potentially wrong steps, checking consistency with problem constraints, and auditing the final derivation or answer. This pattern is consistent with the mechanism of LRS. Since LRS does not use behavior annotations, behavior-specific vectors, or predefined behavior triggers, these shifts should not be interpreted as explicit behavior selection. Rather, they suggest that reward-guided latent optimization changes fragile reasoning steps in ways associated with implicitly promoting more frequent helpful cognitive behaviors through latent-state optimization.

Case Study: AIME25 Q2 — Counting with Strict Constraints

Question Nine players choose chocolate, vanilla, or strawberry; each flavor appears and the counts satisfy $c > v > s$. Count assignments modulo 1000.

Base Error The base sets up $c + v + s = 9$ but keeps only $(5, 3, 1)$, giving $\binom{9}{5,3,1} = 504$. It rejects $(6, 2, 1)$ and $(4, 3, 2)$ despite $6 > 2 > 1$ and $4 > 3 > 2$.

LRS Correction LRS revisits the enumeration and keeps all valid partitions: $(6, 2, 1)$, $(5, 3, 1)$, and $(4, 3, 2)$, with counts 252, 504, and 1260; hence $2016 \equiv 16 \pmod{1000}$.

Diagnostic Pathway Early reward and confidence interventions occur during count branching and after steering, the trace rechecks discarded cases and strict ordering.

Interpretation The case illustrates *Structured Decomposition*, *Constraint Grounding*, and *Course Correction*. SAE dimension names remain post-hoc priors, not deterministic causal labels.

Figure 4: LRS repairs an incomplete enumeration by recovering the missed valid partitions.

Case study. The compact case below illustrates the mechanism suggested by the aggregate analyses. The base reasoning commits to an incomplete enumeration and fails to revisit excluded cases, leading to an incorrect answer. In contrast, the LRS-steered trace receives early latent interven-

tions when the reward model signal indicates that enumeration steps are fragile. After reward-guided optimization at the fragile enumeration stage, cognitive behavior relevant SAE latent dimensions associated with algebraic execution and variable extraction become more active, and the trace later revisits the missing cases to produce a more complete solution. Additional qualitative cases with fuller diagnostics are provided in Appendix D and Figure 12. Together with the aggregate behavior-frequency analysis, this case provides along with more cases D provide post-hoc evidence for RQ3: LRS-steered reasoning traces are associated with implicitly promoting more frequent useful cognitive behaviors.

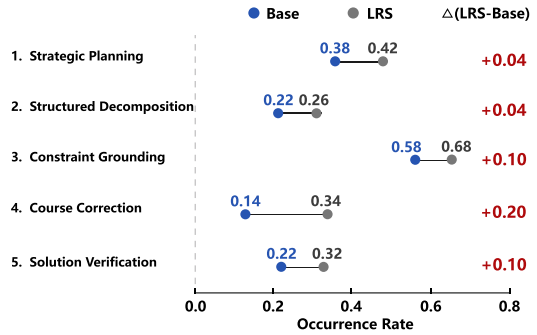

Figure 5: Post-hoc analysis shows that LRS-steered traces more often exhibit course correction, constraint grounding, and solution verification, with behavior labels used only for analysis.

4.5 Additional Analyses: Stability, Interpretability, and Efficiency

We include 3 supporting analyses to examine the stability, interpretability, and practical cost of LRS.

Selective intervention and steering strength. Table 1 shows that LRS BASIC improves some datasets but degrades others, indicating that reward-guided gradients should not be applied indiscriminately. Full LRS uses the reward and confidence gate as a selective repair mechanism, and Figure 6 shows that stronger updates do not monotonically improve accuracy and moderate intervention is more stable.

SAE interpretability. Table 3 summarizes max-activating-context interpretations of the 10 SAE dimensions on MATH-500. These names provide post-hoc priors for analyzing latent changes, not deterministic mappings from dimensions to behaviors.

Dim.	Max Act.	Interpreted Pattern
z_0	0.218	Geometric / structural modeling
z_1	0.287	Theorem invocation / logical branching
z_2	0.477	Algebraic flow / step execution
z_3	0.582	Symbolic math / formatting
z_4	0.552	Variable initialization / constant extraction
z_5	0.438	Property definition / formula grounding
z_6	0.405	Conclusion / answer consolidation
z_7	0.256	Constraint checking / boundary validation
z_8	0.455	Strategy selection / planning
z_9	0.283	Complexity / asymptotic reasoning

Table 3: Compact interpretation of SAE latent dimensions from max-activating contexts on MATH-500.

Metric	Base	LRS
Avg. generated tokens	2595	2596
Avg. wallclock / problem (s)	115.3	156.2
Avg. generation cost (ms / token)	44.5	60.2
Slowdown ratio	1.00×	1.35×
Steered tokens (%)	—	27.9
Unsteered tokens (%)	—	72.1
Avg. steering triggers / problem	—	725
Triggers on correct answers	—	526
Triggers on wrong answers	—	924
Avg. $\|\Delta z\|$ per steered token	—	1.24
SAE latent dimension	—	10

Table 4: Inference efficiency of BASE and LRS on 200 problems from AIME24, AIME25, AMC23, and INEQMATH.

Inference efficiency. Table 4 reports the inference overhead of LRS. Average wall-clock time increases from 115.3s to 156.2s per problem (1.35×), while the reward and confidence gate skips 72.1% of tokens and steers 27.9%.

5 Conclusion

We presented **Latent Reward Steering**, an adaptive inference-time framework that improves LLM reasoning by applying reward-guided optimization to SAE latent states. LRS learns a latent reward

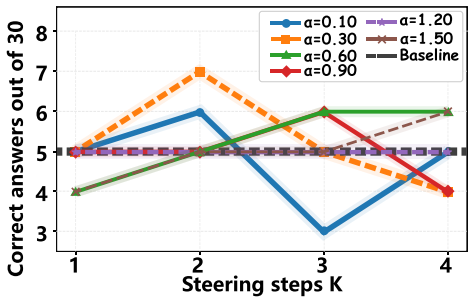

Figure 6: Steering-strength sensitivity on AIME24. Accuracy peaks under moderate updates rather than increasing monotonically and the dashed line indicates standard decoding.

model from successful and unsuccessful reasoning traces, uses reward gradients to correct fragile token-level states, and applies a reward–confidence gate to keep intervention selective. Experiments on OPEN-REASONER-7B and OPEN-REASONER-1.5B show consistent gains across reasoning benchmarks without updating model weights. Reward-score separation, post-hoc behavior analysis, and case studies further suggest that LRS-steered traces are associated with more frequent useful cognitive behaviors. These findings support latent-state optimization as a promising direction for inference-time reasoning improvement.

Limitations

This work has several limitations as follows.

- **Inference overhead.** LRS requires reward-model evaluation and latent-gradient updates during decoding. Selective gating reduces unnecessary interventions, but LRS remains slower than standard decoding.
- **Training–inference mismatch.** The reward model is trained on latent reasoning sequences but queried on local token-level states for efficiency. Future work could explore prefix-level or memory-augmented reward estimation.

Ethical considerations

LRS performs inference-time latent steering based on a learned reward signal, which may introduce risks if the reward model is misaligned or poorly calibrated. In such cases, steering could amplify undesirable reasoning patterns, increase overconfidence in incorrect answers, or produce behavioral shifts that are difficult to interpret. Since latent-space interventions are less transparent than explicit prompting, their effects in open-ended and safety-sensitive scenarios remain uncertain. Our evaluation is limited to mathematical and scientific reasoning benchmarks, and does not fully characterize these risks. Future use of LRS should involve reward-model auditing, broader safety evaluation, and monitoring for unintended changes in model cognitive behavior.

References

Art of Problem Solving. 2024a. 2024 aime i problems and solutions. https://artofproblemsolving.com/wiki/index.php/2024_AIME_I. Accessed: 2026-05-24.

Art of Problem Solving. 2024b. 2024 aime ii problems and solutions. https://artofproblemsolving.com/wiki/index.php/2024_AIME_II. Accessed: 2026-05-24.

Art of Problem Solving. 2025a. 2025 aime i problems and solutions. https://artofproblemsolving.com/wiki/index.php/2025_AIME_I. Accessed: 2026-05-24.

Art of Problem Solving. 2025b. 2025 aime ii problems and solutions. https://artofproblemsolving.com/wiki/index.php/2025_AIME_II. Accessed: 2026-05-24.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Runjin Chen, Zhenyu Zhang, Junyuan Hong, Souvik Kundu, and Zhangyang Wang. 2025. Seal: Steerable reasoning calibration of large language models for free. In *Conference on Language Modeling (COLM)*.

Weixin Chen, Yuhan Zhao, Jingyuan Huang, Zihe Ye, Clark Mingxuan Ju, Tong Zhao, Neil Shah, Li Chen, and Yongfeng Zhang. 2026. Memrec: Collaborative memory-augmented agentic recommender system. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL 2026)*.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. 2023. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2024. Chain-of-verification reduces hallucination in large language models. In *Findings of the association for computational linguistics: ACL 2024*, pages 3563–3578.

Hanwen Du, Yuxin Dong, and Xia Ning. 2025. Latent thinking optimization: Your latent reasoning language model secretly encodes reward signals in its latent thoughts. *arXiv preprint arXiv:2509.26314*.

Zeyu Gan, Yun Liao, and Yong Liu. 2025. Rethinking external slow-thinking: From snowball errors to probability of correct reasoning. *arXiv preprint arXiv:2501.15602*.

Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D. Goodman. 2025. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. In *Conference on Language Modeling (COLM)*.

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 arXiv:2103.03874*.

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. 2026. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *Advances in Neural Information Processing Systems*, 38:162239–162262.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2023. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*.

Can Jin, Jiakang Li, Rui Wu, Eddy Zhang, and Dimitris N Metaxas. 2026. Weak critics make strong learners: On-policy critique distillation for scalable oversight. *arXiv preprint arXiv:2606.00424*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yutaka Iwasawa. 2022. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*.

math-ai. 2025. Amc 2023 dataset. https://huggingface.co/datasets/math-ai/amc23. Accessed: 2026-05-24.

Ning Miao, Yee Whye Teh, and Tom Rainforth. 2023. Selfcheck: Using llms to zero-shot check their own step-by-step reasoning. *arXiv preprint arXiv:2308.00436*.

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*.

Jiayi Sheng, Luna Lyu, Jikai Jin, Tanglin Xia, Alex Gu, James Zou, and Pan Lu. 2026. Solving inequality proofs with large language models. *Advances in Neural Information Processing Systems*, 38.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L. Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, and 3 others. 2024. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. Transformer Circuits Thread. Online.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. 2023. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*.

Gladys Tyen, Hassan Mansoor, Victor Cărbune, Yuanzhu Peter Chen, and Tony Mak. 2024. Llms cannot find reasoning errors, but can correct them given the error location. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13894–13908.

Constantin Venhoff, Iván Arcuschin, Philip Torr, Arthur Conmy, and Neel Nanda. 2025a. Base models know how to reason, thinking models learn when. Under review as a conference paper at ICLR 2026; arXiv:2510.07364.

Constantin Venhoff, Iván Arcuschin, Philip Torr, Arthur Conmy, and Neel Nanda. 2025b. Understanding reasoning in thinking language models via steering vectors. In *Workshop on Reasoning and Planning for Large Language Models*.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 2609–2634.

Yadong Wang, Haodong Chen, Yu Tian, Chuanxing Geng, Dong Liang, and Xiang Chen. 2026. Beyond dense states: Elevating sparse transcoders to active operators for latent reasoning. *arXiv preprint arXiv:2602.01695*.

Jake Ward, Chuqiao Lin, Constantin Venhoff, and Neel Nanda. 2025. Reasoning-finetuning repurposes latent representations in base models. *arXiv preprint arXiv:2507.12638*.

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

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large language models are better reasoners with self-verification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2550–2575.

Wencheng Ye, Xiaoyang Yuan, Yi Bin, Hengyu Jin, Liang Peng, Pengpeng Zeng, and Heng Tao Shen. 2026. Riser: Orchestrating latent reasoning skills for adaptive activation steering. *arXiv preprint arXiv:2601.09269*.

Zhenyu Zhang, Xiaoxia Wu, Zhongzhu Zhou, Qingyang Wu, Yineng Zhang, Pragaash Ponnusamy, Harikaran Subbaraj, Jue Wang, Shuaiwen Leon Song, and Ben

Athiwaratkun. 2025. Understanding and steering the cognitive behaviors of reasoning models at test-time. *arXiv preprint arXiv:2512.24574*.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H Chi, Quoc V Le, and Denny Zhou. 2024. Take a step back: Evoking reasoning via abstraction in large language models. In *International Conference on Learning Representations*, volume 2024, pages 20279–20316.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and 1 others. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. 2023. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*.

CoT Prompt

Let's solve this problem step by step, problem: `<problem>`

Figure 7: Zero-shot chain-of-thought prompt used as a baseline.

A Training and Experimental Details

Models and intervention layer. We use OPEN-REASONER-7B as the primary model and OPEN-REASONER-1.5B as a smaller-model comparison within the same family. The base model parameters are never updated.

Zero-shot decoding and scoring. All benchmarks are evaluated in a zero-shot setting with greedy decoding and batch size 1. During generation, the model receives only the original problem statement, without few-shot demonstrations or task-specific initial reasoning prompts. All final answers are parsed from full output and parsed answers are then scored with the benchmark-specific evaluator: math-style benchmarks use rule-based or math-verification scoring where applicable, while multiple-choice tasks use extracted option letters against the gold answer.

Prompt baselines. The CoT baseline uses the same problem statement with a fixed instruction prefix: For few-shot prompting, we use fixed five-example prompts before the test problem. Math-style datasets use elementary algebra, LCM, geometry, counting, and summation examples. GPQA-DIAMOND uses five multiple-choice science examples and INEQMATH uses five inequality examples covering AM-GM, triangle inequality, and algebraic nonnegativity. The full prompt templates are shown below.

Reward model training. The reward model is trained on sparse latent reasoning sequences obtained by encoding the model's hidden activations through the pretrained SAE (Section 3.1). For each generation, we keep only the reasoning segment starting from the model's "think" marker, discarding the prompt prefix, and pair the sequence with a binary final-answer correctness label that is repeated over all positions for token-level supervision. Architecturally, the reward model is a lightweight Transformer encoder operating directly on d_z-dimensional SAE latents: it applies a LayerNorm on the raw SAE input (essential, since unnor-

Five-Shot Math Prompt

Here are 5 example problems and their solutions.
Example 1. Problem: If $3x + 2 = 11$, what is the value of x^2? Solution: From $3x + 2 = 11$ we get $3x = 9$, so $x = 3$. Therefore $x^2 = 9$. Final answer: $\boxed{9}$
Example 2. Problem: What is the smallest positive integer divisible by both 6 and 8? Solution: Prime factorize: $6 = 2 \cdot 3$ and $8 = 2^3$. So $\mathrm{lcm}(6, 8) = 2^3 \cdot 3 = 24$. Final answer: $\boxed{24}$
Example 3. Problem: A right triangle has legs of length 5 and 12. What is the length of the hypotenuse? Solution: By the Pythagorean theorem, $c = \sqrt{5^2 + 12^2} = \sqrt{169} = 13$. Final answer: $\boxed{13}$
Example 4. Problem: How many ways are there to choose 3 books from a shelf of 7 distinct books? Solution: This is $\binom{7}{3} = \frac{7 \cdot 6 \cdot 5}{3 \cdot 2 \cdot 1} = 35$. Final answer: $\boxed{35}$
Example 5. Problem: What is the sum of the first 100 positive integers? Solution: Using $\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$ with $n = 100$, the sum is $\frac{100 \cdot 101}{2} = 5050$. Final answer: $\boxed{5050}$
Now solve this problem.
Problem: `<problem>`

Figure 8: Five-shot prompt template for math-style benchmarks.

Five-Shot GPQA Prompt

Here are 5 example multiple-choice problems and their solutions.
Example 1. Problem: A ball is dropped from a height of 20 m. How long does it take to hit the ground? Ignore air resistance, $g = 10\,\mathrm{m/s^2}$. Choices: (A) 1 s, (B) 2 s, (C) 4 s, (D) 5 s. Solution: Using $h = \frac{1}{2}gt^2$, solve $20 = 5t^2$, so $t = 2$. Final answer: $\boxed{B}$
Example 2. Problem: Which of the following has the highest electronegativity? Choices: (A) Carbon, (B) Nitrogen, (C) Oxygen, (D) Fluorine. Solution: Electronegativity increases across a period and decreases down a group. Fluorine has the highest value. Final answer: $\boxed{D}$
Example 3. Problem: Which organelle is responsible for ATP production via oxidative phosphorylation in eukaryotic cells? Choices: (A) Nucleus, (B) Ribosome, (C) Mitochondrion, (D) Endoplasmic reticulum. Solution: The mitochondrion houses the electron transport chain and ATP synthase. Final answer: $\boxed{C}$
Example 4. Problem: A 2 kg object moves at 3 m/s. What is its kinetic energy? Choices: (A) 3 J, (B) 6 J, (C) 9 J, (D) 12 J. Solution: $KE = \frac{1}{2}mv^2 = \frac{1}{2}(2)(3)^2 = 9$ J. Final answer: $\boxed{C}$
Example 5. Problem: What is the pH of a 10^{-3} M HCl solution? Choices: (A) 1, (B) 3, (C) 7, (D) 11. Solution: HCl fully dissociates, so $[\mathrm{H}^+] = 10^{-3}$ M and $\mathrm{pH} = 3$. Final answer: $\boxed{B}$
Now solve this problem.
Problem: `<problem>`

Figure 9: Five-shot prompt template for GPQA-Diamond.

Five-Shot IneqMath Prompt

Here are 5 example inequality problems and their solutions.
Example 1. Problem: Find the largest constant C such that $x^2 + y^2 \geq Cxy$ for all real x, y. Solution: $x^2 + y^2 - 2xy = (x - y)^2 \geq 0$, so $C = 2$ is tight. Final answer: $\boxed{2}$
Example 2. Problem: For all positive reals a, b, determine the relation between $a + b$ and $2\sqrt{ab}$. Solution: By AM-GM, $\frac{a+b}{2} \geq \sqrt{ab}$, so $a + b \geq 2\sqrt{ab}$. Final answer: $\boxed{\geq}$
Example 3. Problem: Find the smallest constant C such that $|x + y| \leq C(|x| + |y|)$ for all real x, y. Solution: By the triangle inequality, $C = 1$ works and is tight. Final answer: $\boxed{1}$
Example 4. Problem: For positive reals a, b, c, determine the relation between $a^2 + b^2 + c^2$ and $ab + bc + ca$. Solution: $a^2 + b^2 + c^2 - ab - bc - ca = \frac{1}{2}((a - b)^2 + (b - c)^2 + (c - a)^2) \geq 0$. Final answer: $\boxed{\geq}$
Example 5. Problem: Find the largest constant C such that $(a + b)^2 \geq Cab$ for all positive reals a, b. Solution: By AM-GM, $a + b \geq 2\sqrt{ab}$, so $(a + b)^2 \geq 4ab$, with equality at $a = b$. Final answer: $\boxed{4}$
Now solve this problem.
Problem: `<problem>`

Figure 10: Five-shot prompt template for IneqMath.

malized SAE activations are often near-zero and easily dominated by positional encoding), a linear projection to hidden size $d = 128$, sinusoidal positional encoding, two Transformer encoder blocks (4 attention heads, feedforward width $4d$, dropout 0.1), and an MLP head (Linear–ReLU–Dropout–Linear–Sigmoid) producing a per-position probability $p_{i,t} \in (0, 1)$. We optimize with AdamW (learning rate 5×10^{-4}, weight decay 10^{-4}), binary cross-entropy loss, gradient clipping at max-norm 1.0, and batch size 1 (one variable-length trace per step). To counter class imbalance between correct and incorrect trajectories, we apply weighted random sampling, with each trace sampled with probability inversely proportional to its class frequency. We train for 30 epochs and select the checkpoint with the lowest training loss for inference-time steering. No behavior labels, GPT-4o annotations, or predefined cognitive categories are used at any stage and the only supervision is final-answer correctness.

Steering variants and gate. LRS (BASIC) applies reward-guided latent steering without the reward and confidence gate. Full LRS uses the gate to selectively intervene: steering is triggered when the reward score is below the reward threshold, or when the reward score is above threshold but the previous-token decoding confidence, measured as the last-token maximum probability, is below the confidence threshold. In the experiments below, most conditions are instantiated as reward < 0.9 or reward ≥ 0.9 with last-token maximum probability < 0.72, except for configurations whose thresholds are listed separately in Table 5.

B Behavior Annotation Protocol

We use GPT-4o only for post-hoc diagnostic annotation of generated reasoning traces. The same fixed prompt is applied to base and LRS outputs. These annotations are never used for reward-model training, steering-vector construction, or gate triggering.

Annotation prompt. We apply the same fixed GPT-4o prompt for both base and LRS traces, also we use the following behavior criteria:

- **Strategic Planning:** mark true if the trace selects a method, theorem, or overall strategy before detailed computation and explains why it is appropriate.
- **Structured Decomposition:** mark true if the trace breaks the problem into cases, subproblems, lemmas, branches, or explicitly named intermediate goals.
- **Constraint Grounding:** mark true if the trace actively uses problem constraints to check domains, exclude invalid solutions, validate boundary conditions, or restrict the search space.
- **Course Correction:** mark true if the trace detects an error, contradiction, missing case, or uncertainty and then revises the computation, switches methods, or redirects the solution path.
- **Solution Verification:** mark true if the trace substitutes an intermediate result or final answer back into the original problem, or otherwise checks that the derived answer satisfies the required conditions.

Occurrence and improvement. For each trace, a behavior has occurrence value 1 if GPT-4o marks it as present at least once and 0 otherwise. The occurrence rate is the average of this indicator over the matched trace set. Behavior improvement is computed as the LRS occurrence rate minus the Base occurrence rate. Because the labels are post-hoc diagnostics, they support process-level interpretation rather than direct claims that LRS explicitly controls predefined behaviors.

C Case Diagnostic Notes

This appendix clarifies how to read the diagnostics used in the qualitative cases. The case-study boxes report steering-event counts, early trigger positions, and the largest corrected SAE dimensions. These diagnostics are derived from per-question SAE steering traces and support a cautious process-level reading: they show where latent updates were applied and which sparse dimensions changed most, but they do not provide token-level reward values or prove a causal mechanism. We therefore avoid numeric reward claims and interpret dimension names only as post-hoc priors from Table 3.

D Qualitative Case Studies with Reward-Signal Diagnostics

We provide representative examples where LRS changes an initially incorrect single generation into a correct answer. Each case highlights the critical divergence between the base and LRS-steered

Dataset	Model	K	α	Reward	Confidence	Device
AMC23	Open-Reasoner-7B	1	1.400	0.9	0.72	RTX A4500
AMC23	Open-Reasoner-1.5B	4	0.300	0.9	0.72	RTX A6000
AIME24	Open-Reasoner-7B	2	0.295	0.9	0.72	RTX A4500
AIME24	Open-Reasoner-1.5B	4	0.900	0.9	0.72	RTX A4500
AIME25	Open-Reasoner-7B	3	1.320	0.9	0.72	RTX A4500
AIME25	Open-Reasoner-1.5B	2	0.400	0.9	0.72	RTX A6000
GPQA Diamond	Open-Reasoner-7B	4	1.150	0.9	0.72	RTX A5000
GPQA Diamond	Open-Reasoner-1.5B	3	1.000	0.9	0.72	RTX A5000
IneqMath	Open-Reasoner-7B	2	0.700	0.9	0.72	RTX A5000
IneqMath	Open-Reasoner-1.5B	2	1.100	0.9	0.72	RTX A6000
MATH-500	Open-Reasoner-7B	1	1.400	0.8	0.69	RTX A4500
MATH-500	Open-Reasoner-1.5B	1	0.100	0.9	0.72	RTX A6000

Table 5: Experimental configurations for different datasets and models. K denotes the number of latent optimization steps, and α denotes the step size. The reward and confidence columns report the corresponding gate thresholds. The device column reports the GPU used for each dataset–model configuration.

Behavior	**Trace-level criterion**	**Associated SAE dimensions**
Strategic Planning	Selects a method or theorem and explains the rationale before detailed computation, e.g., choosing a Diophantine strategy before algebraic execution.	z_8 meta-cognitive strategy; z_1 logical branching.
Structured Decomposition	Breaks the problem into independent branches, such as case splits, named lemmas, or formally defined intermediate sub-problems.	z_1 theorem invocation; z_0 structural decomposition.
Constraint Grounding	Uses problem constraints to guide or limit the solution process, such as checking domains, excluding invalid solutions, or verifying boundary conditions.	z_7 constraint processing; z_6 deductive conclusion.
Course Correction	Detects a problem during reasoning and adjusts direction, such as switching methods after a contradiction or revising a flawed assumption.	z_8 meta-cognitive re-planning; z_7 violation detection; z_1 logical branching.
Solution Verification	Substitutes the final answer back into the original problem or checks whether the derived result satisfies the required conditions.	z_7 constraint processing; z_6 answer consolidation.

Table 6: Cognitive behavior categories used for matched base-vs.-LRS trace analysis. Associated SAE dimensions are used only as interpretability priors for analyzing Δz, not as deterministic behavior mappings.

trajectories. These examples cover different failure modes, including combinatorial overcounting, missing cases, incorrect constraint handling, scientific concept confusion, and algebraic simplification errors. The highlighted failure modes are used only for post-hoc analysis. LRS itself does not rely on behavior labels or predefined steering directions. Excerpts are shortened for readability while preserving the original reasoning error, the corrected reasoning step, and the final answer. SAE dimension interpretations are post-hoc priors and should not be read as deterministic labels, especially for non-mathematical tasks such as GPQA.

Cross-case latent dimension patterns. Across all five cases, LRS reward-guided steering consistently up-regulates concrete algebraic manipulation dimensions, especially z_2 for algebraic step execution and z_4 for variable initialization, while down-regulating dimensions associated with premature conclusion formation or complexity evaluation, such as z_6 and z_9. The dimension z_2 appears as a top-corrected dimension in all five cases, suggesting that the latent reward signal primarily promotes step-by-step algebraic reasoning over abstract planning or premature answer selection. This pattern holds across math competition tasks, graduate-level science questions, and inequality-proof settings.

GPT-4o Behavior Annotation Prompt

Input. You will receive two fields: `problem`, the original problem statement, and `reasoning_trace`, the generated solution.

Task. For each behavior below, decide whether it appears at least once anywhere in the reasoning trace. Do not judge final-answer correctness except when the trace explicitly verifies a result against the original problem.

Output. Return only a JSON object with one key per behavior. Each key must contain a Boolean field `present` and a short `evidence` string. If a behavior is absent, set `present` to `false` and use an empty evidence string.

Schema. `{StrategicPlanning: {present: bool, evidence: str}, StructuredDecomposition: {present: bool, evidence: str}, ConstraintGrounding: {present: bool, evidence: str}, CourseCorrection: {present: bool, evidence: str}, SolutionVerification: {present: bool, evidence: str}}`

Figure 11: Prompt used for post-hoc GPT-4o behavior annotation.

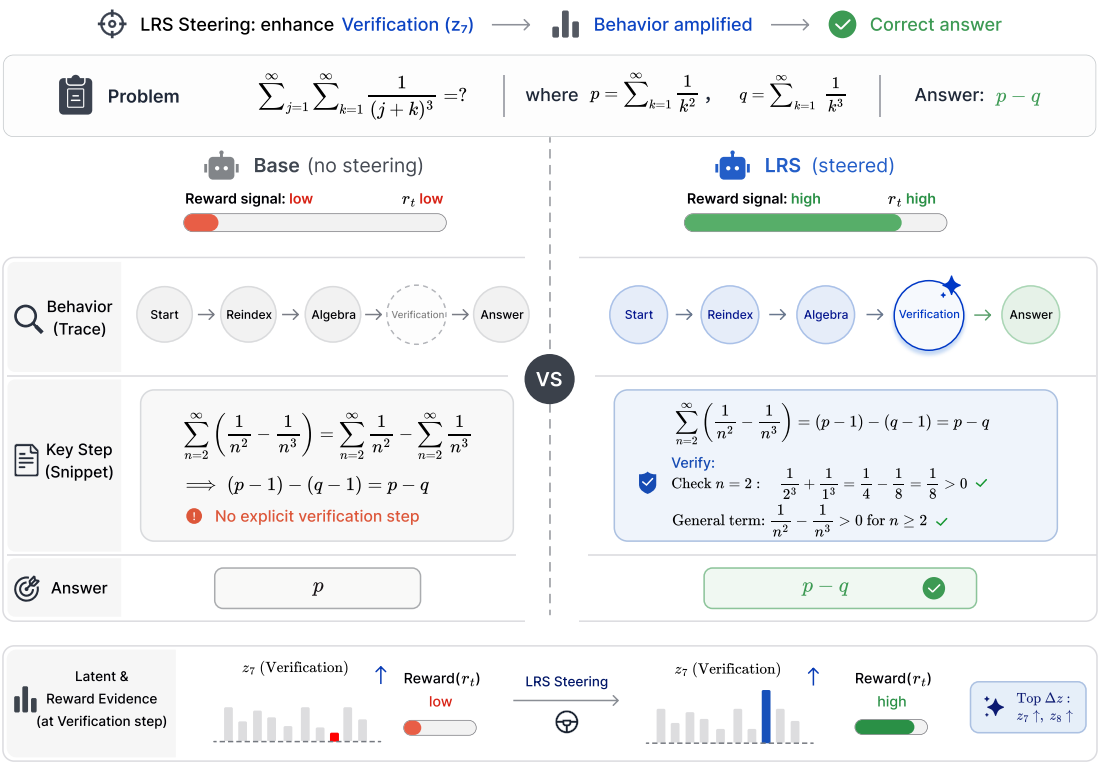

Figure 12: Qualitative example where LRS steers an incorrect reasoning trace toward verification and a correct answer.

Case Study 1: AMC23 Q38 — Combinatorics (Subset Counting)

Question Determine the number of nonempty subsets B of $\{0, 1, 2, \ldots, 12\}$ such that the number of elements in B is equal to the least element of B.

Base Output The base model lets k denote the least element of B and correctly requires $|B| = k$ with $k \geq 1$. It then chooses $k-1$ additional elements from $\{k, k+1, \ldots, 12\}$, which contains $13-k$ elements, yielding $\binom{13-k}{k-1}$. Summing for $k = 1$ to 7: $1 + 11 + 45 + 84 + 70 + 21 + 1 = 233$. The error is that k itself is already required to be in B, so the remaining $k-1$ elements must be chosen from $\{k+1, \ldots, 12\}$, which has $12-k$ elements, not from $\{k, \ldots, 12\}$.

LRS Steered Output The LRS-steered trace uses the correct selection pool: $\binom{12-k}{k-1}$. "The subset B must include k and $k-1$ other elements, all of which must be greater than k. The elements of B can be chosen from the set $\{k+1, k+2, \ldots, 12\}$. The number of elements in this set is $12-k$." Summing for $k = 1$ to 6: $1 + 10 + 36 + 56 + 35 + 6 = 144$.

Key Difference Base uses $\binom{13-k}{k-1}$, which includes k in the selection pool and causes overcounting. LRS uses $\binom{12-k}{k-1}$, correctly excluding k.

Steering Info 184 steering events and the first recorded steering events occur at generation steps 24, 59, and 69. Top corrected dimensions: $z_2 \uparrow$ (algebraic execution), $z_4 \uparrow$ (variable extraction), $z_9 \downarrow$ (complexity evaluation).

Reward-Signal Diagnostic The base reasoning trace begins to fail when it constructs the counting pool $\{k, \ldots, 12\}$ and thereby counts the required minimum element k twice. The reward and confidence gate is associated with early interventions before the final summation, suggesting that the local latent state around the combinatorial setup was treated as fragile. In the steered trace, the solution grounds the constraint that k is already included in B and changes the pool to $\{k+1, \ldots, 12\}$. This change is consistent with *Constraint Grounding* and *Course Correction*, rather than a behavior label explicitly supplied to LRS.

Latent-Change Interpretation The largest average latent changes up-regulate z_2 and z_4, which Table 3 associates with algebraic execution and variable initialization, while down-regulating z_9, associated with complexity evaluation. This pattern is consistent with the steered trace correcting the counting setup and avoiding an overcomplicated selection pool. These dimension names are post-hoc interpretability priors, not deterministic causal labels.

Figure 13: AMC23 Q38: LRS corrects an overcount caused by including k in the selection pool.

Case Study 2: AIME25 Q2 — Counting (Multinomial Partitions)

Question Nine players each choose one of three flavors: chocolate, vanilla, or strawberry. Each flavor is chosen by at least one player, and the number choosing chocolate is strictly greater than the number choosing vanilla, which is strictly greater than the number choosing strawberry. Find the number of valid assignments modulo 1000.

Base Output Setting $c > v > s \geq 1$ with $c + v + s = 9$, the base model enumerates:

- $s = 1$: finds $(5, 3, 1)$ as valid, but rejects $(6, 2, 1)$ by incorrectly claiming $v > s$ fails, although $2 > 1$ holds.
- $s = 2$: tests $(4, 3, 2)$ but rejects it, writing "not valid since $c > v$", which misapplies the constraint.

Only the single partition $(5, 3, 1)$ is retained: $\binom{9}{5,3,1} = 504$, so the answer is 504.

LRS Steered Output The LRS-steered reasoning trace correctly enumerates all three valid partitions:

- $(6, 2, 1)$: valid since $6 > 2 > 1$, $\binom{9}{6,2,1} = 252$.
- $(5, 3, 1)$: valid, $\binom{9}{5,3,1} = 504$.
- $(4, 3, 2)$: valid since $4 > 3 > 2$, $\binom{9}{4,3,2} = 1260$.

Total: $252 + 504 + 1260 = 2016$. Remainder modulo 1000: 16.

Key Difference Base misses two of three valid partitions due to incorrect constraint checks. LRS correctly enumerates all three.

Steering Info 134 steering events and the first recorded steering events occur at generation steps 14, 15, and 18. Top corrected dimensions: $z_2 \uparrow$ (algebraic execution), $z_3 \uparrow$ (symbolic reasoning), $z_0 \downarrow$ (geometric modeling).

Reward-Signal Diagnostic The base reasoning trace starts to fail during case enumeration: it rejects valid partitions such as $(6, 2, 1)$ and $(4, 3, 2)$ despite the strict ordering constraints being satisfied. The early reward and confidence gate activations suggest that the latent state during the enumeration phase was flagged as locally fragile. The steered trace revisits the missing cases and evaluates all valid multinomial partitions before applying the modulo operation. This pattern is associated with *Structured Decomposition*, *Constraint Grounding*, and *Course Correction*.

Latent-Change Interpretation The dominant changes up-regulate z_2 and z_3, associated with algebraic execution and symbolic formatting, while down-regulating z_0, associated with structural modeling. This is consistent with the steered trace moving from an incomplete structural enumeration to explicit validation of each multinomial case. These dimension names are post-hoc interpretability priors, not deterministic causal labels.

Figure 14: AIME25 Q2: LRS recovers the missed valid partitions and the correct remainder.

Case Study 3: AMC23 Q25 — Complex Number Algebra

Question Define the operation $u \otimes v = ac + bdi$, where $u = a + bi$ and $v = c + di$. Find $|z|^2$ given that $z \otimes z = z^2 + 40$.

Base Output Both models set $z = x + yi$ and correctly derive the real-part constraint $y^2 = 40$. For the imaginary part, they reach $y^2 i = 2xy$. The base model argues: "The left side $y^2 i$ is purely imaginary, and the right side $2xy$ is a real number. For these to be equal ... this implies $2xy = 0$ and $y^2 = 0$. However, we already found that $y^2 = 40$, which means $y \neq 0$. Therefore, the only way ... is if $x = 0$." With $x = 0$ and $y^2 = 40$, it concludes $|z|^2 = 0 + 40 = 40$.

LRS Steered Output The LRS-steered reasoning trace handles the same equation correctly: "This implies $y^2 = 2xy$. If $y \neq 0$, we can divide both sides by y: $y = 2x$." Combined with $y^2 = 40$, this gives $(2x)^2 = 40 \Rightarrow x^2 = 10$. Therefore, $|z|^2 = x^2 + y^2 = 10 + 40 = 50$.

Key Difference Base treats $y^2 i = 2xy$ as requiring both real and imaginary parts to vanish independently, forcing $x = 0$. LRS correctly compares the imaginary coefficients to obtain $y = 2x$.

Steering Info 247 steering events and the first recorded steering events occur at generation steps 9, 37, and 48. Top corrected dimensions: $z_2 \uparrow$ (algebraic execution), $z_9 \downarrow$ (complexity evaluation), $z_4 \uparrow$ (variable extraction).

Reward-Signal Diagnostic The base reasoning trace starts to fail when it separates the imaginary-part equation incorrectly and forces $x = 0$ instead of comparing imaginary coefficients. The reward and confidence gate is associated with interventions near the algebraic manipulation stage, suggesting low reliability for the local equation-handling state. In the steered trace, the solution compares imaginary coefficients directly, derives $y = 2x$, and then completes the norm computation. This shift is consistent with *Constraint Grounding* and *Course Correction*.

Latent-Change Interpretation The largest average changes up-regulate z_2 and z_4, associated with algebraic execution and variable extraction, while down-regulating z_9, associated with complexity evaluation. This pattern is consistent with the steered trace replacing an invalid coefficient split with direct manipulation of the real and imaginary constraints. These dimension names are post-hoc interpretability priors, not deterministic causal labels.

Figure 15: AMC23 Q25: LRS fixes the imaginary-part constraint and obtains $|z|^2 = 50$.

Case Study 4: GPQA Diamond Q39 — Chemistry (Separation Science)

Question A synthetic organic chemist tells a colleague: "My compounds are on top of each other." What is the second chemist most likely referring to?

(A) The compounds have similar polarities.

(B) The compounds are bonding through non-covalent interactions.

(C) The compounds have similar optical rotations.

(D) The compounds have similar boiling points.

Base Output The base model considers both chromatography and distillation but ultimately favors distillation: "The most likely reason for the compounds being 'on top of each other' ... is that they have similar boiling points. This would make it difficult to separate them using distillation." Answer: D.

LRS Steered Output The LRS-steered trace identifies the chromatography context: "Similar polarities (option A) is a broad property that can affect the solubility and separation of compounds, making it a plausible explanation for the observed issue. The phrase 'on top of each other' suggests that the compounds are not well-separated or are difficult to distinguish from one another." Answer: A.

Key Difference Base interprets "on top of each other" through the lens of distillation and boiling points. LRS connects the phrase to chromatographic separation and polarities.

Steering Info 653 steering events and the first recorded steering events occur at generation steps 2, 3, and 6. Top corrected dimensions: $z_3 \uparrow$ (symbolic reasoning), $z_2 \uparrow$ (algebraic execution), $z_0 \downarrow$ (geometric modeling).

Reward-Signal Diagnostic The base reasoning trace begins to drift when it interprets "on top of each other" through a distillation frame and selects boiling points, rather than grounding the phrase in chromatographic separation. The very early gate activations suggest that the reward and confidence signal treated the initial conceptual framing as unstable. The steered trace redirects the interpretation toward compounds co-eluting or poorly separating on a chromatographic medium, leading to similar polarities. This change is associated with *Strategic Planning* at the conceptual-framing level and *Constraint Grounding* in the domain-specific clue.

Latent-Change Interpretation The largest average changes up-regulate z_3 and z_2 and down-regulate z_0. Although Table 3 names these dimensions using math-heavy max-activating contexts, for this non-mathematical GPQA case they should be read only as post-hoc latent-change summaries. The observed pattern is consistent with an early reframing of the domain clue, not with a deterministic "algebraic" mechanism.

Figure 16: GPQA Diamond Q39: LRS reframes the clue as chromatographic separation.

Case Study 5: IneqMath Q84 — Inequality Proof (AM-GM)

Question Determine the correct relation for

$$S = a\sqrt[3]{1+b-c} + b\sqrt[3]{1+c-a} + c\sqrt[3]{1+a-b}$$

versus 1, given $a + b + c = 1$ with $a, b, c > 0$. Options include $\leq$, $=$, $\geq$, etc.

Base Output Testing $a = b = c = \frac{1}{3}$ gives $S = 1$. The base model then attempts to rewrite the terms using the constraint:

$$1 + b - c = a + 2b, \qquad 1 + c - a = b + 2c, \qquad 1 + a - b = 2a + c.$$

This simplification is incorrect. For example, $1 + b - c$ does not simplify to $a + 2b$ under $a + b + c = 1$. The model then attempts AM-GM on the wrong expressions and ultimately concludes: "Given these inequalities, the only consistent relationship is $S = 1$." No valid option letter is extracted.

LRS Steered Output The LRS-steered reasoning trace avoids the false simplification and reasons directly: "Let $x = 1+b-c$, $y = 1+c-a$, $z = 1+a-b$. Note that $x+y+z = 3$. By the AM-GM inequality: $\sqrt[3]{1+b-c} \leq \frac{x+y+z}{3} = 1$. Therefore $a\sqrt[3]{1+b-c} \leq a$, and similarly for the other terms. Summing: $S \leq a + b + c = 1$." Combined with $S = 1$ at the symmetric point, the answer is A: $\leq$.

Key Difference Base makes an incorrect algebraic simplification and then conjectures equality without proof. LRS applies AM-GM directly to the original cube-root terms and correctly establishes $S \leq 1$.

Steering Info 559 steering events and the first recorded steering events occur at generation steps 4, 5, and 6. Top corrected dimensions: $z_4 \uparrow$ (variable extraction), $z_2 \uparrow$ (algebraic execution), $z_9 \downarrow$ (complexity evaluation).

Reward-Signal Diagnostic The base reasoning trace starts to fail when it rewrites $1 + b - c$ as $a + 2b$, creating an invalid algebraic premise for the later AM-GM argument. The early reward and confidence gate activations are associated with this fragile symbolic setup, suggesting that the latent reward model treated the local manipulation state as fragile. The steered trace avoids the false simplification, defines auxiliary variables directly from the original cube-root terms, and applies AM-GM to the valid quantities. This behavior is consistent with *Constraint Grounding* and *Course Correction*.

Latent-Change Interpretation The largest average changes up-regulate z_4 and z_2, associated with variable extraction and algebraic execution, while down-regulating z_9, associated with complexity evaluation. This pattern is consistent with the steered trace avoiding the false rewrite and operating directly on the original cube-root terms. These dimension names are post-hoc interpretability priors, not deterministic causal labels.

Figure 17: IneqMath Q84: LRS avoids a false simplification and applies AM-GM correctly.