RETHINKING REASONING PATHS AS PHASE-STRUCTURED TRAJECTORIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models often improve problem-solving performance by generating multi-step reasoning paths, yet how to analyze the hidden states along these paths remains unclear. Existing approaches typically assign each intermediate state the final-answer correctness label and train probes across heterogeneous questions. We argue that this protocol obscures reasoning dynamics in two ways: (1) correctness prediction can exploit question-level variation rather than path quality, and (2) states aligned by absolute step indices may correspond to different functional phases of reasoning. In this work, we propose to view reasoning paths as phase-structured trajectories within fixed questions. We instantiate this view as **PAIR**, short for **P**hase-**A**ligned **I**ntra-question **R**easoning. PAIR samples multiple trajectories for each question, maps variable-length paths into shared relative phases based on normalized trajectory progress, and compares successful and unsuccessful trajectories only within the same question and phase. This yields phase-specific path-quality directions that better isolate path-quality signals from question-level variation. Empirically, we find that standard across-question correctness probes lose much of their predictive power under within-question evaluation, suggesting that these probes partly rely on question-level information. PAIR improves within-question trajectory ranking and Best-of-N trajectory selection across models and benchmarks. Phase-wise steering further shows that the learned directions can change generation outcomes, providing causal evidence that they capture trajectory-relevant information.

1 INTRODUCTION

Large language models can improve their problem-solving performance by generating multi-step reasoning paths before producing final answers. This behavior is most explicit in chain-of-thought prompting (Wei et al., 2022), where models are encouraged to write intermediate reasoning steps, and has been shown to improve performance on mathematical reasoning (Cobbe et al., 2021), symbolic reasoning, and code generation tasks (Chen et al., 2021). The accuracy gains from this style of generation are substantial, but the internal mechanism by which it succeeds or fails remains opaque. As a model generates a reasoning path, *what information about the eventual correctness is encoded in its hidden states, and how does this information evolve along the trajectory?*

Recent work has increasingly analyzed language model reasoning through internal representations, using hidden-state probes, representation geometry, and activation directions to uncover task-relevant or correctness-related signals (Burns et al.; Li et al., 2023; He et al., 2026). In many such analyses, intermediate hidden states are treated as pointwise representations labeled by the final outcome, e.g., $y = \mathbb{I}[\hat{a} = a^*]$, and a probe or scorer is trained to predict final correctness from each state (Fig 1a).

However, such across-question correctness prediction can conflate path quality with question-level variation (Fig 1b). When examples are pooled across heterogeneous questions, the correctness label depends not only on the generated trajectory, but also on properties of the question being solved. This makes early correctness signals hard to interpret: strong prediction before substantial reasoning has occurred may reflect question-level information rather than path-level progress.

A second issue concerns how intermediate states are compared along the trajectory. Final correctness is a trajectory-level outcome, but pointwise probing assigns this same label to every intermediate state along the path. Recent work has begun to study reasoning as a structured trajectory, showing

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

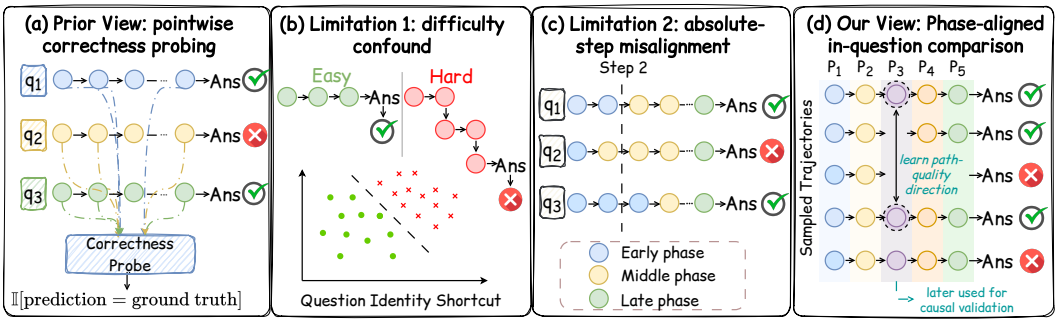


Figure 1: Overview of our motivation. (a) Prior probing labels intermediate states by final-answer correctness across different questions. (b) Across-question prediction can exploit question identity rather than path quality. (c) Absolute step indices can misalign reasoning phases across questions. (d) We instead align trajectories by phase and compare paths within the same question.

that step-specific activations can form separable subspaces and that correct and incorrect trajectories diverge at later stages (Sun et al., 2026). These results suggest that reasoning has meaningful temporal geometry. However, existing trajectory analyses often align states by explicit step markers or absolute step indices. This creates a structural mismatch: "Step 2" can mean setup in one solution and the main calculation in another (Fig 1c).

In this work, we view reasoning paths as phase-structured trajectories within fixed questions (Fig. 1d). A useful comparison should hold the question fixed, so that question-level information cannot serve as the main shortcut, and should compare states that occupy comparable positions within their trajectories. Under this view, the relevant question is not whether a hidden state predicts final correctness across a heterogeneous dataset, but whether one trajectory state is better than another for the same question and at the same relative phase.

We instantiate this view as PAIR, short for **P**hase-**A**ligned **I**ntra-question **R**easoning. PAIR samples multiple reasoning trajectories for each question, maps variable-length paths to a shared set of phase slots, and constructs phase-matched comparisons between successful and unsuccessful trajectories from the same question. From these comparisons, PAIR learns phase-specific directions that rank successful states above unsuccessful states. We then use these directions for analysis and steering, testing whether they can actively redirect generation.

Empirically, we find that standard across-question correctness signals weaken substantially under within-question evaluation, suggesting that part of their apparent strength comes from question-level information. We further show that phase alignment reveals trajectory-quality structure that is obscured by absolute-step comparisons. The learned phase-specific directions improve within-question trajectory ranking and Best-of-N trajectory selection. Finally, phase-wise steering shows that these directions can change generation outcomes, providing causal evidence that they capture trajectory-relevant information.

Our contributions are as follows:

- We diagnose key ambiguities in pointwise correctness analysis, showing that across-question probes can capture question-level variation and that absolute step indices can misalign reasoning stages.
- We introduce PAIR, a phase-aligned intra-question framework that compares successful and unsuccessful trajectories from the same question at matched relative phases to learn path-quality directions.
- We demonstrate that PAIR improves trajectory ranking and Best-of-N selection, and use phase-wise steering to provide causal evidence that the learned directions can affect reasoning generation.

2 Related Work

Reasoning Paths and Multi-trajectory Inference. Chain-of-thought prompting improves language-model reasoning by eliciting intermediate steps before the final answer (Wei et al., 2022; Kojima et al., 2022). Beyond single-path generation, self-consistency samples multiple reasoning paths for the same question and aggregates their answers (Wang et al., 2022), while tree- and graph-based methods cast reasoning as search over candidate thoughts or states (Yao et al., 2023; Besta et al., 2024). Other work evaluates or supervises intermediate reasoning through process supervision and step-level verification (Lightman et al., 2023). These methods exploit path structure to improve answer selection or reasoning performance; in contrast, we use multiple trajectories for the same question to study how reasoning quality is represented in hidden states.

Representation Analysis of Reasoning Trajectories. Recent work analyzes model behavior through internal representations, including latent knowledge, truthfulness, confidence, and task-relevant structure (Burns et al.; Azaria & Mitchell, 2023), as well as representation geometry and activation directions for characterizing or intervening on behavior (Li et al., 2023; Park et al., 2023). In reasoning tasks, intermediate hidden states have been shown to encode signals related to final-answer correctness, self-verification, and reasoning progress before the answer is generated (Zhang et al., 2025; Liu et al., 2025). Most closely related to our work, Sun et al. (2026) view reasoning as a structured trajectory in representation space, showing that activations near explicit reasoning markers form step-specific subspaces and that correct and incorrect trajectories may diverge over generation. Our work shares this trajectory-level view but addresses a different comparison problem: absolute step indices are not always comparable across trajectories, since the second step of one solution may be problem setup while the second step of another may already contain the main computation. We therefore align variable-length trajectories by relative phase and compare states within the same question and phase.

Correctness Signals and Activation Steering. Although correctness-related information can often be decoded from intermediate hidden states (Zhang et al., 2025), such signals are hard to interpret when examples are pooled across heterogeneous questions. A predictor trained across questions may capture question-level information, such as difficulty or prior solvability, rather than the quality of a particular reasoning trajectory. PAIR instead compares successful and unsuccessful trajectories within the same question and aligned phase, avoiding across-question correctness predictors. For causal validation, we apply the resulting phase-specific directions through activation steering. Prior work shows that hidden-state directions can steer model behavior at inference time (Turner et al., 2023), and contrastive activation methods further learn behavior-changing directions from paired examples (Panickssery et al., 2023).

3 PAIR: Phase-Aligned Intra-question Reasoning

In this section, we introduce PAIR, a phase-aligned intra-question framework for analyzing and steering reasoning trajectories. PAIR is built on the premise that reasoning paths should not be compared as absolute-step sequences across heterogeneous questions. Instead, paths should be compared within the same question and at comparable relative phases of generation.

As shown in Fig. 2, PAIR consists of three stages. First, for each question, we sample multiple reasoning trajectories and align their hidden states into a fixed set of relative phase slots. Second, within each phase, we construct successful–unsuccessful trajectory pairs from the same question and learn a phase-specific path-quality direction through a pairwise ranking objective. Third, we use the learned directions to steer hidden states during generation, thereby testing whether phase-specific path-quality signals can causally redirect reasoning trajectories.

3.1 Problem Setup

Let q denote a question with ground-truth answer a_q^*. For each question, we sample multiple reasoning trajectories from the model. The i-th trajectory is written as

$$\tau_{q,i} = \left(x_{q,i,1:T_{q,i}}, H_{q,i,1:T_{q,i}}, \hat{a}_{q,i}\right), \tag{1}$$

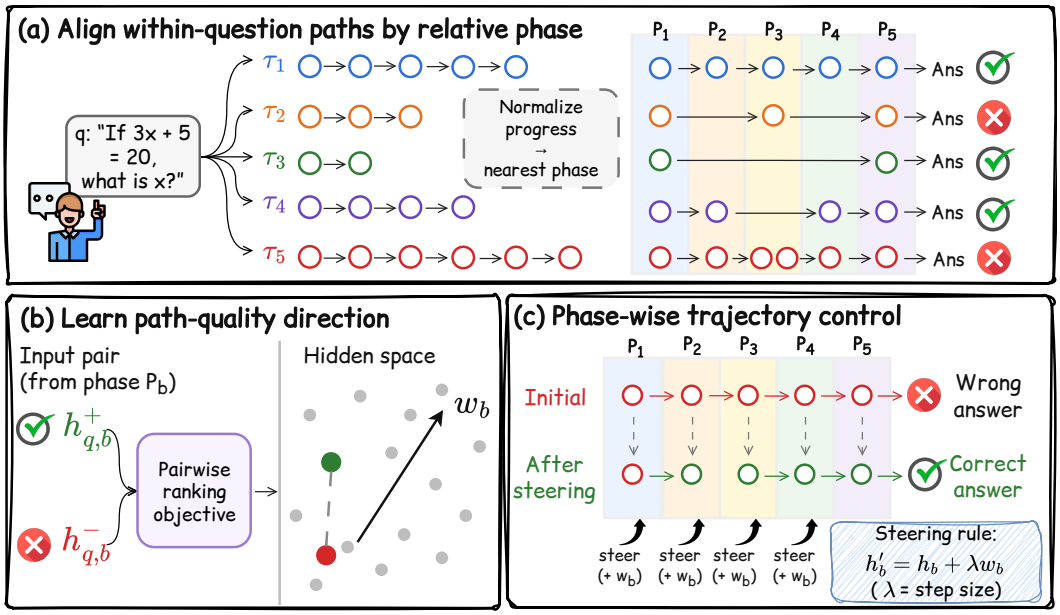


Figure 2: Overview of PAIR. (a) For each question, PAIR samples multiple reasoning trajectories and aligns variable-length paths into a fixed set of relative phase slots. (b) Within each phase, PAIR constructs successful–unsuccessful trajectory pairs from the same question and learns a phase-specific path-quality direction using a pairwise ranking objective. (c) The learned directions are used for phase-wise activation steering, where hidden states are shifted toward successful trajectory directions during generation.

where $x_{q,i,1:T_{q,i}}$ are generated tokens, $H_{q,i,1:T_{q,i}}$ are token-level hidden states, and $\hat{a}_{q,i}$ is the final answer. The trajectory-level outcome is

$$y_{q,i} = \mathbb{I}\left[\hat{a}_{q,i} = a_q^*\right]. \tag{2}$$

As discussed in Sec. 1, directly using $y_{q,i}$ as a label for every intermediate state can conflate local reasoning quality with final-answer correctness and question-level variation. Our goal is therefore not to predict final correctness from isolated hidden states. Instead, we ask whether one trajectory state is better than another when the question and the reasoning phase are matched.

After phase alignment, let $h^{(\ell)}_{q,i,b}$ denote the layer-ℓ representation of trajectory $\tau_{q,i}$ at phase b. For two trajectories sampled from the same question, we define the phase-wise path-quality preference:

$$h^{(\ell)}_{q,i,b} \succ h^{(\ell)}_{q,j,b} \quad \text{if} \quad y_{q,i} = 1,\ y_{q,j} = 0. \tag{3}$$

Thus, final correctness is used to form preferences between trajectories matched by question and phase, rather than to label individual hidden states.

3.2 Phase Alignment of Reasoning Trajectories

PAIR first converts sampled reasoning trajectories into phase-aligned step representations. This stage has two components: extracting step-level features from each trajectory and mapping variable-length step sequences to shared relative phases.

Step-level trajectory representation. For each question q, we sample m reasoning trajectories under the same prompting and decoding configuration. The trajectory and outcome notation follows Eqs. 1–2. Following Sun et al. (2026), we extract step-level representations using explicit reasoning markers. We first detect step markers, such as "Step 1", numbered-list markers, and discourse markers such as "First"; the full marker set is given in Appendix A.4.1. These markers indicate the beginning of reasoning steps.

Let $p_{q,i,k}$ denote the token position of the k-th detected step marker in trajectory $\tau_{q,i}$, and let $c_{q,i}$ denote the position of the conclusion marker. If a trajectory contains $L_{q,i}$ reasoning steps, we represent step j using the r tokens immediately before the next step marker; the final step is represented using the r tokens before the conclusion marker:

$$h^{(\ell)}_{q,i,j} = \begin{cases} \mathrm{Pool}\Big(\Big\{H^{(\ell)}_{q,i,t} : p_{q,i,j+1} - r \le t < p_{q,i,j+1}\Big\}\Big), & j < L_{q,i}, \\ \mathrm{Pool}\Big(\Big\{H^{(\ell)}_{q,i,t} : c_{q,i} - r \le t < c_{q,i}\Big\}\Big), & j = L_{q,i}, \end{cases} \tag{4}$$

where $H^{(\ell)}_{q,i,t}$ is the token-level hidden state at layer ℓ, and $\mathrm{Pool}(\cdot)$ denotes window aggregation. This gives each trajectory a step-level hidden-state sequence $\{h^{(\ell)}_{q,i,j}\}_{j=1}^{L_{q,i}}$.

Relative phase alignment. Since trajectories may contain different numbers of reasoning steps, absolute step indices are not directly comparable. PAIR maps each trajectory to K relative phase slots. We set K to the mode of the step counts in the training trajectories:

$$K = \mathrm{mode}\left(\{L_{q,i} : q \in \mathcal{D}_{\mathrm{train}},\ i = 1, \dots, m\}\right). \tag{5}$$

For a trajectory with $L_{q,i}$ steps, step j is assigned to its nearest relative phase:

$$b_{q,i,j} = 1 + \mathrm{round}\left(\frac{(j-1)(K-1)}{\max(L_{q,i}-1, 1)}\right), \quad b_{q,i,j} \in \{1, \dots, K\}. \tag{6}$$

This maps the first step to P_1 and the final step to P_K, while intermediate steps are placed according to normalized progress. As illustrated in Fig. 2(a), shorter trajectories may skip some phase slots, whereas longer trajectories may assign multiple steps to the same phase.

For each trajectory and phase, we collect the step representations assigned to that phase:

$$\mathcal{H}^{(\ell)}_{q,i,b} = \left\{h^{(\ell)}_{q,i,j} : b_{q,i,j} = b\right\}. \tag{7}$$

The resulting phase-indexed sets are used to construct within-question comparisons in the next stage.

3.3 Learning Phase-specific Path-quality Directions

After phase alignment, PAIR learns one path-quality direction for each phase.

For phase b, let $\mathcal{H}^{(\ell)}_{q,i,b}$ be the set of step representations from trajectory $\tau_{q,i}$ assigned to phase b, as defined in Eq. 7. We collect successful and unsuccessful phase-b states for question q:

$$\mathcal{H}^{+}_{q,b} = \bigcup_{i:\, y_{q,i}=1} \mathcal{H}^{(\ell)}_{q,i,b}, \qquad \mathcal{H}^{-}_{q,b} = \bigcup_{i:\, y_{q,i}=0} \mathcal{H}^{(\ell)}_{q,i,b}. \tag{8}$$

For each question q, PAIR constructs within-question preference pairs:

$$\mathcal{P}_{q,b} = \left\{(h^+, h^-) : h^+ \in \mathcal{H}^{+}_{q,b},\ h^- \in \mathcal{H}^{-}_{q,b}\right\}. \tag{9}$$

Each pair states that, for the same question and same phase, the state from a successful trajectory should receive a higher path-quality score than the state from an unsuccessful trajectory.

For each phase b, we fit a linear scorer after phase-specific preprocessing. Let $\phi_b(\cdot)$ denote the preprocessing map fitted on the training states for phase b. The phase-b score is

$$s_b(h) = \tilde{w}_b^{\top} \phi_b(h) + \beta_b, \tag{10}$$

where $\tilde{w}_b$ is the learned direction in the preprocessed space. PAIR optimizes the pairwise logistic ranking objective:

$$\mathcal{L}_b = -\sum_{q} \sum_{(h^+, h^-) \in \mathcal{P}_{q,b}} \log \sigma\left(s_b(h^+) - s_b(h^-)\right) + \lambda \|\tilde{w}_b\|_2^2. \tag{11}$$

Optimizing Eq. 11 yields a phase-specific path-quality direction, which we map back to the original hidden-state space and normalize as w_b for analysis and steering.

3.4 PHASE-WISE TRAJECTORY STEERING

The learned directions are used as causal interventions on the reasoning trajectory. As illustrated in Fig. 2(c), PAIR modifies the hidden state according to the direction associated with the current reasoning phase.

Let w_b be the normalized raw-space path-quality direction for phase b. During generation, let $H_t^{(\ell)}$ denote the hidden state at layer ℓ and decoding position t. Let $b_t \in \{1, \ldots, K\}$ be the phase assigned to the current position by the steering schedule. PAIR applies the intervention

$$\widetilde{H}_t^{(\ell)} = H_t^{(\ell)} + \lambda_t w_{b_t}, \tag{12}$$

where λ_t is the intervention strength at position t. The modified state $\widetilde{H}_t^{(\ell)}$ is passed to the remaining layers, with all model parameters fixed.

Rather than using a constant strength, we use the phase-specific score to adapt the intervention magnitude. Let $\widetilde{s}_b(h)$ denote the learned phase-b path-quality score, and let τ_b be a reference score estimated from successful training trajectories at phase b. The intervention strength is

$$\lambda_t = \min\left\{\lambda_{\max},\ \alpha\left[\tau_{b_t} - \widetilde{s}_{b_t}\left(H_t^{(\ell)}\right)\right]_+\right\}, \tag{13}$$

where $[x]_+ = \max(x, 0)$, α controls the global steering scale, and $\lambda_{\max}$ clips overly large updates. Thus, PAIR pushes a state only when its phase-specific score falls below the reference level, and the clipping term prevents unstable interventions.

We evaluate two intervention schedules. In single-phase steering, Eq. 12 is applied only at one selected phase. In progressive steering, the intervention follows the trajectory phase and uses the corresponding direction w_{b_t} as generation advances. The threshold construction, clipping value, and phase-schedule implementation are given in Appendix A.4.3.

4 EXPERIMENTS AND RESULTS

We structure our experiments to test the main claim that reasoning trajectories require phase-aligned within-question comparison. We first diagnose the limitations of standard across-question correctness probing, showing that it can conflate path quality with question-level variation and step misalignment (Sec. 4.2). We then evaluate whether PAIR learns phase-specific path-quality directions that improve within-question ranking of successful and unsuccessful trajectories (Sec. 4.3). Finally, we use phase-wise activation steering to test whether these directions can causally redirect generation rather than merely correlate with final correctness (Sec. 4.4).

4.1 EXPERIMENTAL SETUP

Models and benchmarks. We evaluate on three reasoning benchmarks: GSM8K Cobbe et al. (2021), MATH500 Lightman et al. (2023), and the logical_deduction_three_objects task from Big-Bench Hard (BBH) Suzgun et al. (2022). We use four open-weight models: LLaMA-3.1-8B-Instruct Meta (2024), Qwen3-4B Team (2025), Gemma3-4B Team et al. (2025), and DeepSeek-R1-Distill-Llama-8B DeepSeek-AI (2025). For GSM8K, we train probes on 1,000 questions from the official training split and evaluate on 500 questions from the official test split. For MATH and BBH, we use a fixed 70/30 question-level split, sampled once and reused across all methods and models. All evaluations are conducted on held-out test questions, so trajectories from the same question never appear in both training and test sets.

Trajectory sampling and labeling. For each question, we sample 32 reasoning trajectories from the evaluated model with temperature 1.0 and top-$p = 0.95$. A trajectory is labeled correct if its final extracted answer matches the ground truth, using dataset-specific extraction rules; unparsable outputs are treated as incorrect. Sampling multiple trajectories per question lets us compare successful and unsuccessful paths under a shared question context, controlling for question-level difficulty.

Table 1: Across-question and within-question evaluation of correctness probes. For each benchmark, we report across-question (AQ) and within-question (WQ) AUROC and balanced accuracy, together with $\Delta = \text{WQ} - \text{AQ}$. AQ/WQ entries are formatted as AQ/**WQ**.

Model	GSM8K (Cobbe et al., 2021)				MATH (Lightman et al., 2023)				BBH (Suzgun et al., 2022)			
	AUC	Δ	Balanced Acc.	Δ	AUC	Δ	Balanced Acc.	Δ	AUC	Δ	Balanced Acc.	Δ
LLaMA-3.1-8B-Instruct	0.72/**0.56**	-0.16	0.66/**0.51**	-0.15	0.65/**0.49**	-0.16	0.64/**0.50**	-0.14	0.72/**0.53**	-0.19	0.66/**0.51**	-0.15
Qwen3-4B	0.72/**0.53**	-0.19	0.73/**0.64**	-0.09	0.64/**0.51**	-0.13	0.60/**0.57**	-0.03	0.77/**0.53**	-0.24	0.76/**0.51**	-0.25
Gemma3-4B	0.65/**0.50**	-0.15	0.63/**0.52**	-0.11	0.68/**0.54**	-0.14	0.61/**0.58**	-0.03	0.71/**0.51**	-0.20	0.71/**0.41**	-0.30
DeepSeek-R1-Distill-Llama-8B	0.63/**0.62**	-0.01	0.62/**0.59**	-0.03	0.65/**0.51**	-0.14	0.64/**0.56**	-0.08	0.68/**0.51**	-0.17	0.66/**0.57**	-0.09
Average	0.68/**0.55**	-0.13	0.66/**0.57**	-0.10	0.66/**0.51**	-0.14	0.62/**0.55**	-0.07	0.72/**0.52**	-0.20	0.70/**0.50**	-0.20

Representation extraction. Unless otherwise specified, we probe the final transformer layer, where prior work has found the strongest signals for truthfulness and task outcomes (Azaria & Mitchell, 2023; Burns et al.; Li et al., 2023). For each trajectory, we locate step markers and average the hidden states of the four tokens immediately preceding each marker to form a step-level representation. These step-level representations are shared across all hidden-state scoring methods we compare, including final-token probing, pooled correctness probing, absolute-step alignment, normalized-step alignment, and our phase-aligned PAIR method.

4.2 Diagnosing Ambiguities in Pointwise Correctness Analysis

We first test whether pointwise correctness prediction remains reliable after controlling for question identity. For each sampled trajectory, we extract the hidden states from a four-token window around the position closest to 15% relative progress and use their mean as the representation. We then train a linear predictor using the trajectory-level correctness label $y_{q,i}$. Additional training details are given in Appendix A.4.3.

We report the standard across-question evaluation (AQ), which tests whether the predictor can predict final correctness on held-out questions. We contrast it with within-question evaluation (WQ), which ranks correct and incorrect trajectories sampled for the same question:

$$\text{WQ-AUC} = \frac{1}{|\mathcal{Q}_\pm|} \sum_{q \in \mathcal{Q}_\pm} \frac{1}{|\mathcal{P}_q||\mathcal{N}_q|} \sum_{i \in \mathcal{P}_q} \sum_{j \in \mathcal{N}_q} \mathbb{I}\left[s(x^+_{q,i}) > s(x^-_{q,j})\right], \tag{14}$$

where $\mathcal{P}_q$ and $\mathcal{N}_q$ denote the successful and unsuccessful trajectories for question q, respectively, and $\mathcal{Q}_\pm$ contains questions with at least one trajectory from each group. Balanced accuracy is computed under the same within-question protocol.

Table 1 shows that AQ performance consistently overestimates the strength of correctness signals. Across models and benchmarks, AUROC and balanced accuracy drop sharply under WQ evaluation. For example, the average BBH AUROC decreases from 0.72 to 0.52, and balanced accuracy decreases from 0.70 to 0.50. This gap indicates that across-question correctness predictors capture substantial question-level information, such as difficulty or solvability, rather than only trajectory-level progress.

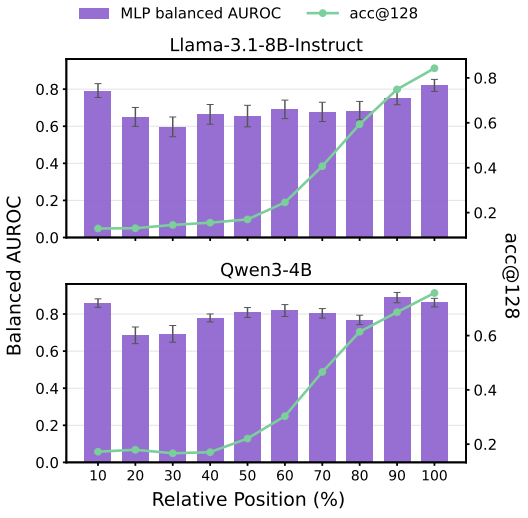


Figure 3: Balanced AUROC of correctness prediction and forced-answer accuracy across relative trajectory positions.

To test whether early correctness signals reflect answerability, we compare correctness prediction with forced-answer accuracy along the trajectory. Following prior work on early correctness signals (Zhang et al., 2025), we train a two-layer MLP predictor on window-aggregated hidden states at different relative positions, using final-answer correctness as the label. Following the forced-answer extraction protocol of Boppana et al. (2026), we estimate answerability by truncating each trajectory at the same relative position, appending "The answer is", and sampling

Table 2: Evaluation of trajectory-quality estimation and Best-of-32 trajectory selection. We highlight the best value in **green** **with bold text** and the second best value in blue.

Model	Method	GSM8K (Cobbe et al., 2021)		MATH (Lightman et al., 2023)		BBH (Suzgun et al., 2022)	
		WQ-AUC ↑	BoN@32 ↑	WQ-AUC ↑	BoN@32 ↑	WQ-AUC ↑	BoN@32 ↑
LLaMA-3.1 -8B-Instruct Meta (2024)	Pre-Ans token	0.71	83.3	**0.83**	**46.6**	0.65	84.0
	Pooled correctness	0.66	84.6	0.69	36.0	0.51	77.3
	Absolute-step	0.63	81.6	0.63	36.6	0.57	81.3
	Normalized-step	0.67	85.3	0.71	40.0	0.58	68.0
	PAIR	**0.76**	**87.3**	0.77	**46.6**	**0.74**	**92.0**
Qwen3-4B (Team, 2025)	Pre-Ans token	0.48	58.7	0.79	83.3	0.84	93.3
	Pooled correctness	0.61	74.7	0.54	74.6	0.65	92.0
	Absolute-step	0.59	55.3	0.52	73.3	0.61	85.0
	Normalized-step	0.68	**80.7**	0.60	76.0	0.62	94.6
	PAIR	**0.91**	**80.7**	**0.89**	**84.7**	**0.86**	**97.3**
Gemma3-4B (Team et al., 2025)	Pre-Ans token	0.60	89.3	0.60	68.7	0.67	94.6
	Pooled correctness	0.48	89.3	0.53	61.3	0.51	92.0
	Absolute-step	0.55	83.3	0.49	55.3	0.55	94.6
	Normalized-step	0.57	88.0	0.54	66.7	0.62	**96.0**
	PAIR	**0.61**	**95.0**	**0.74**	**75.3**	**0.75**	**96.0**
DeepSeek-R1- Distill-Llama-8B DeepSeek-AI (2025)	Pre-Ans token	0.65	86.0	**0.87**	84.6	0.92	**100.0**
	Pooled correctness	0.60	88.0	0.53	72.0	0.56	81.3
	Absolute-step	0.58	84.3	0.63	74.6	0.50	69.3
	Normalized-step	0.57	86.6	0.66	80.0	0.84	**100.0**
	PAIR	**0.75**	**89.3**	**0.87**	**86.7**	**0.94**	**100.0**

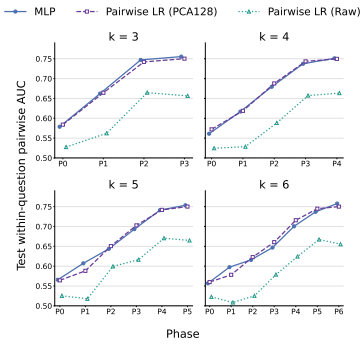


Figure 4a. WQ-AUC across phases.

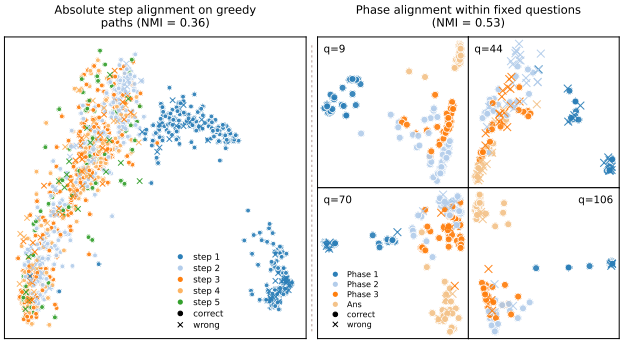


Figure 4b. Absolute-step versus phase alignment.

$k = 128$ completions with temperature 0.8. We define $\mathrm{acc@}k = c/k$, where c is the number of completions that produce the correct answer.

Fig. 3 shows the results on GSM8K for Llama-3.1-8B-Instruct and Qwen3-4B. The MLP predictor achieves high balanced AUROC at very early positions, consistent with prior findings, while acc@128 remains close to zero. This indicates that early representations can predict final correctness even when the prefix is not yet answerable. Across relative positions, AUROC first decreases and then increases, whereas acc@128 rises as the trajectory approaches the final answer. The late increase in AUROC is expected because later prefixes contain more information needed to recover the answer. *The early decrease may reflect a reduced influence of question-level information as generation moves away from the prompt.* Together with Table 1, this result supports the need to control question identity when analyzing pointwise correctness signals.

4.3 Effectiveness of Phase-Aligned Path-Quality Directions

We first examine how path-quality separability changes across aligned phases. For each phase, we train a phase-specific scorer and evaluate it by pairwise AUROC (Eq. 14) on held-out questions, using $K \in \{3, 4, 5, 6\}$ phase bins. As shown in Fig. 4a, AUROC generally increases in later phases, indicating that later states better separate successful and unsuccessful trajectories. This is consistent with prior findings that correct and incorrect trajectories diverge over generation (Sun et al., 2026). Because the final phase gives the strongest and most stable separation, Table 2 uses the final-phase scorer for trajectory selection. We also find that raw pairwise LR is weaker, while PCA-compressed LR approaches the MLP, suggesting that linear directions are effective after reducing high-dimensional nuisance variation. We therefore use the PCA-based linear scorer as the main PAIR estimator.

Table 2 evaluates trajectory scoring for within-question ranking and Best-of-N selection. In BoN@32, the model samples 32 reasoning trajectories for each question, and the scorer selects the highest-scoring trajectory as the final output. We compare PAIR with four pointwise correctness baselines trained on greedy trajectories with final-answer correctness labels. Pre-Ans token uses the represen-

Table 3: Phase-wise steering results across benchmarks. Baseline denotes generation without intervention. Single P_b applies the phase-b direction only at phase P_b, while PAIR applies phase-specific directions progressively along the trajectory. We report final-answer accuracy, wrong-to-correct and correct-to-wrong flip rates, and generated-token length after steering.

Intervention	GSM8K				MATH				BBH			
	Acc.	$W\to C$	$C\to W$	Tok.	Acc.	$W\to C$	$C\to W$	Tok.	Acc.	$W\to C$	$C\to W$	Tok.
Baseline	79.3	–	–	$234_{\pm 70}$	42.0	–	–	$383_{\pm 129}$	74.6	–	–	$146_{\pm 28}$
Single P_1	83.3	32.2	3.3	$242_{\pm 71}$	47.3	12.6	4.7	$399_{\pm 181}$	76.0	57.8	17.9	$167_{\pm 25}$
Single P_2	82.6	25.8	2.5	$233_{\pm 67}$	44.7	6.9	3.2	$386_{\pm 121}$	70.6	5.2	7.1	$187_{\pm 34}$
Single P_3	80.0	3.2	0	$234_{\pm 70}$	42.0	2.3	3.2	$387_{\pm 170}$	69.3	36.8	19.6	$205_{\pm 66}$
Single P_4	79.3	0.0	0.0	$234_{\pm 70}$	42.7	1.1	0	$386_{\pm 123}$	76.0	10.5	1.8	$198_{\pm 54}$
Single P_5	79.3	0.0	0.0	$234_{\pm 70}$	42.0	0.0	0.0	$383_{\pm 129}$	72.0	0.0	3.6	$169_{\pm 26}$
PAIR (Ours)	84.6	38.7	3.3	$244_{\pm 74}$	47.3	12.6	4.7	$398_{\pm 181}$	76.0	57.8	17.9	$167_{\pm 25}$

tation immediately before the answer as a late-stage reference. Pooled correctness uses the mean four-token representation near 25% relative progress, following prior work on early correctness signals (Zhang et al., 2025). Absolute-step uses the third-step representation, reflecting step-index-based comparison (Sun et al., 2026). Normalized-step uses the step representation closest to 80% relative progress. PAIR achieves the best or competitive WQ-AUC and BoN@32 across most settings, showing that phase-aligned within-question training provides a stronger trajectory-ranking signal than baselines. Pre-Ans token is also strong in several cases, but it is taken very close to the final answer, where answer information is often already recoverable. The relatively strong Normalized-step baseline further suggests that relative trajectory position matters, even without full phase-aligned training.

Figure 4b visualizes the effect of phase alignment. When hidden states are grouped by absolute step indices on greedy paths, only the first step forms a clear cluster, and later steps are mixed, with an average NMI of 0.36. This reflects that the same absolute step can correspond to different reasoning roles across questions. After aligning trajectories within each fixed question by relative phase, phase groups become more separated, increasing the average NMI to 0.53.

4.4 Phase-Specific Steering Redirects Reasoning Trajectories

We use phase-wise activation steering as a causal validation of the learned path-quality directions. Given a generated trajectory, we intervene at the corresponding reasoning phase by adding the learned direction to the hidden state, following Eq. 12. Table 3 reports steering results on LLaMA-3.1-8B-Instruct across benchmarks, comparing no intervention, single-phase steering, and progressive PAIR steering. Single P_b applies only the direction learned for phase P_b, while PAIR applies phase-specific directions as the trajectory advances. We report final-answer accuracy, wrong-to-correct flips $W\to C$, correct-to-wrong flips $C\to W$, and generated-token length after steering.

The steering results show that PAIR can correct some originally unsuccessful trajectories. On GSM8K, accuracy increases from 79.3 to 84.6, with 38.7% of originally wrong trajectories flipped to correct and only 3.3% of originally correct trajectories flipped to wrong. MATH shows a similar accuracy gain, from 42.0 to 47.3, while BBH shows a smaller net gain because the large $W\to C$ rate is partly offset by a higher $C\to W$ rate. The single-phase results also suggest that early phases are more causally sensitive: steering at P_1 or P_2 is often more effective than steering at later phases. This is consistent with Fig. 4a because readout separability and causal influence measure different aspects of the trajectory. Later phases are easier to classify because they are closer to the final outcome, whereas earlier interventions can affect a longer portion of the remaining reasoning process.

5 Conclusion

We presented a phase-aligned view of reasoning trajectories in LLMs. We showed that standard correctness probes can be ambiguous: across-question evaluation may capture question-level variation, and absolute step indices may compare different reasoning stages. PAIR addresses this by comparing successful and unsuccessful trajectories from the same question at matched phases to learn path-quality directions. Phase-wise steering further shows that these directions can change reasoning outcomes rather than only predict them. These results suggest analyzing reasoning trajectories as structured processes, controlling comparisons by both question identity and relative phase.

AI USE STATEMENT

We used generative AI tools in two ways: to aid and polish the writing of this paper, and to assist with retrieval and discovery of related work. All AI-assisted text was reviewed and revised by the authors, and every work suggested through AI-assisted search was read and verified by the authors before being cited. The authors take full responsibility for the content of this paper.

REPRODUCIBILITY STATEMENT

The appendix gives the prompt templates for all evaluated models, model configurations and hidden-state extraction settings, and dataset statistics with question-level splits and trajectory-sampling configurations. It also documents the rule-based step-marker extraction and the mapping from markers to hidden states, the preprocessing and pairwise logistic-regression procedure for learning phase-specific path-quality directions, and the thresholded phase-wise steering together with its intervention schedules and validation-selected hyperparameters. The training, hyperparameter-selection, and refitting protocols for the pointwise probe baselines are included as well, along with the compute resources required. Combined with the problem setup, the phase-alignment procedure, the steering formulation, and the experimental setup presented in the main paper, these materials cover the key components needed to reproduce our results. We will release our code upon publication.

REFERENCES

Amos Azaria and Tom Mitchell. The internal state of an llm knows when it's lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 967–976, 2023.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 17682–17690, 2024.

Siddharth Boppana, Annabel Ma, Max Loeffler, Raphael Sarfati, Eric Bigelow, Atticus Geiger, Owen Lewis, and Jack Merullo. Reasoning theater: Disentangling model beliefs from chain-of-thought. *arXiv preprint arXiv:2603.05488*, 2026.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision, 2024. *URL https://arxiv. org/abs/2212.03827*.

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

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

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Zhenghao He, Guangzhi Xiong, Bohan Liu, Sanchit Sinha, and Aidong Zhang. Reasoning beyond chain-of-thought: A latent computational mode in large language models, 2026. URL https://arxiv.org/abs/2601.08058.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213, 2022.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36:41451–41530, 2023.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Jiarui Liu, Jivitesh Jain, Mona Diab, and Nishant Subramani. Llm microscope: What model internals reveal about answer correctness and context utilization. *arXiv preprint arXiv:2510.04013*, 2025.

Meta. Llama 3.1: Open foundation and instruction-tuned large language models, 2024. URL https://ai.meta.com/llama/. Llama-3.1-8B-Instruct model.

Nina Panickssery, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. Steering llama 2 via contrastive activation addition. *arXiv preprint arXiv:2312.06681*, 2023.

Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. *arXiv preprint arXiv:2311.03658*, 2023.

Lihao Sun, Hang Dong, Bo Qiao, Qingwei Lin, Dongmei Zhang, and Saravan Rajmohan. Llm reasoning as trajectories: Step-specific representation geometry and correctness signals. *arXiv preprint arXiv:2604.05655*, 2026.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, , and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. Activation addition: Steering language models without optimization. arxiv eprints, pages arxiv–2308. 2023.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

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

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

Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit Panda, Jinyang Li, and He He. Reasoning models know when they're right: Probing hidden states for self-verification, 2025. URL https://arxiv.org/abs/2504.05419.

A APPENDIX

A.1 PROMPT DESIGN

The prompt for each model are listed as below:

Prompt for LLaMA3.1-8B-Instruct

```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>
You are a logical reasoning assistant. Solve the user's question by breaking it down into logical steps.
Finally, provide the answer in the specified format.<|eot_id|>
<|start_header_id|>user<|end_header_id|>
Question: {question}

Please think step by step, and at the end output:
Therefore, the answer is [your answer].'<|eot_id|>
<|start_header_id|>assistant<|end_header_id|>
```

Figure 5: Prompt for LLaMA3.1-8B-Instruct

Prompt for Gemma3-4B

```
You are a logical reasoning assistant. Solve the user's question by breaking it down into logical steps.
Finally, provide the answer in the specified format.
Question: {question}\n

Please think step by step, and at the end output:
Therefore, the answer is [your answer].<eos>
```

Figure 6: Prompt for Gemma3-4B

Prompt for Qwen3-4B

```
<|im_start|>system
You are a helpful assistant. Solve the user's question step by step. Finally, provide the answer in the
specified format.<|im_end|>
<|im_start|>user
{question}
Format your final response as: 'Therefore, the answer is [your answer].'/think<|im_end|>
<|im_start|>assistant
```

Figure 7: Prompt for Qwen3-4B

A.2 MODEL DETAILS

We evaluate four open-weight LLMs with different instruction-tuning and reasoning-training backgrounds. All experiments use the publicly released checkpoints without additional finetuning. Unless otherwise specified, hidden-state features are extracted from the final transformer layer, and steering interventions are applied to the same layer.

A.2.1 LLAMA-3.1-8B-INSTRUCT

Llama-3.1-8B-Instruct is an instruction-tuned model from the Llama 3.1 family (Meta, 2024). We use it as the primary model for both diagnostic analysis and phase-wise steering because it provides stable multi-step reasoning trajectories under chain-of-thought prompting. For all experiments, we use the same decoding and trajectory-sampling settings described in Sec. 4.1.

Prompt for DeepSeek-R1-Distilled-LLaMA-8B

<|begin_of_sentence|>
You are a logical reasoning assistant. Solve the user's question by breaking it down into logical steps. Finally, provide the answer in the specified format.
<|User|>
Question: {question}

Please think step by step, and at the end output:
Therefore, the answer is [your answer].
<|Assistant|><think>

Figure 8: Prompt for DeepSeek-R1-Distilled-LLaMA-8B

A.2.2 QWEN3-4B

Qwen3-4B is an open-weight model from the Qwen3 family (Team, 2025). We include it to test whether the observed trajectory-level patterns hold beyond the Llama model family. In our experiments, Qwen3-4B is evaluated with the same prompting format, trajectory-sampling configuration, and hidden-state extraction procedure as the other models.

A.2.3 GEMMA3-4B

Gemma3-4B is an open-weight model from the Gemma 3 family (Team et al., 2025). We use this model as an additional architecture family for evaluating pointwise correctness analysis, phase-aligned trajectory ranking, and Best-of-N trajectory selection. No model-specific finetuning or calibration is applied.

A.2.4 DEEPSEEK-R1-DISTILL-LLAMA-8B

DeepSeek-R1-Distill-Llama-8B is a distilled reasoning model from the DeepSeek-R1 series (DeepSeek-AI, 2025). We include it to evaluate PAIR on a model with stronger explicit reasoning behavior. The model is evaluated with the same trajectory-sampling and representation-extraction pipeline used for the other models. Because distilled reasoning models tend to generate longer reasoning traces, we report generated-token lengths when evaluating steering interventions.

A.3 DATASET DETAILS

We evaluate on three reasoning benchmarks: GSM8K, MATH-500, and BBH. All splits are made at the question level. For every benchmark, trajectories sampled from the same question are assigned to the same split, so no question appears in both training and test sets.

A.3.1 GSM8K

GSM8K is a grade-school math word-problem benchmark (Cobbe et al., 2021). We use GSM8K as the main benchmark for learning phase-specific path-quality directions because it provides a sufficiently large training split. Specifically, we train on 1,000 questions sampled from the official training split and evaluate on 500 questions sampled from the official test split. For each question, we sample 32 reasoning trajectories with temperature 1.0 and top-$p = 0.95$. A trajectory is labeled correct if the extracted final answer matches the ground-truth answer; otherwise, it is labeled incorrect.

A.3.2 MATH-500

MATH-500 is a 500-problem subset of the MATH benchmark (Lightman et al., 2023). It contains competition-style mathematical reasoning problems and is substantially more difficult than GSM8K. Since MATH-500 does not provide a large official training split for our setting, we use a fixed 70/30 question-level split. The split is sampled once and reused across all models and methods to ensure a consistent comparison. For each question, we sample 32 reasoning trajectories using the same

decoding configuration as GSM8K, and correctness is determined by dataset-specific final-answer extraction.

A.3.3 BBH LOGICAL DEDUCTION

We use the `logical_deduction_three_objects` task from Big-Bench Hard (Suzgun et al., 2022). This task evaluates symbolic and logical reasoning over a small set of objects and constraints. As with MATH-500, we use a fixed 70/30 question-level split and reuse the same split across all methods and models. For each question, we sample 32 reasoning trajectories with temperature 1.0 and top-$p = 0.95$. A trajectory is labeled correct if its extracted final answer matches the ground-truth option or answer string under the task-specific extraction rule.

A.4 IMPLEMENTATION DETAILS

A.4.1 STEP MARKER EXTRACTION

We identify intermediate reasoning steps from the model's generated response using a rule-based *smart marker* procedure. The input is first normalized with the same text cleaning used elsewhere in our pipeline, and step markers are then detected on the cleaned response text.

Candidate step markers. The smart marker extractor considers four classes of candidate boundaries:

1. **Explicit numeric markers at line start**, such as `Step 3:`, `3.`, `3)`, or `(3)`.
2. **Section-style numeric headers** at line start, such as `Part 2:`, `Phase 3`, `Round 4`, etc.
3. **Discourse markers**, such as `First`, `Second`, `Next`, `Then`, `After that`, `Finally`, and `Lastly`. These are only accepted when they occur at a line boundary or at a sentence boundary.
4. **Heading-like lines** ending with a colon, provided that they look like short natural-language headers rather than equations or formatting artifacts.

Filtering heuristics. To avoid spurious boundaries, we apply several filters. Numeric or discourse candidates are kept only if the following text looks like genuine prose rather than a formula, a bare number, or a final-answer statement. We explicitly reject separator/table lines (e.g., rows of dashes), equation-only lines, and lines that contain only symbols or numbers. We also strip common Markdown decoration before testing whether a line looks like a valid heading or step.

Preference for explicit structure. If the response already contains a clear explicit step structure (at least two numbered or `Step-`n markers), we trust that structure and ignore weaker discourse-style transitions inside those numbered steps. Otherwise, we merge numeric, discourse, and heading candidates into a single ordered candidate list.

Step numbering. Candidates are processed from left to right and assigned monotonically increasing step indices. Explicit numeric markers keep their stated index. Discourse ordinals such as `First`, `Second`, and `Third` are mapped to their canonical indices. Other accepted discourse or heading markers are assigned the next available step index. We discard any candidate whose assigned index would be non-monotonic or implausibly large.

Mapping markers to hidden states. After converting accepted character positions to token indices, we represent step k using the hidden state immediately before the marker of step $k{+}1$. Thus, a marker for step $k{+}1$ defines the feature for the preceding step k. In addition, we extract:

- a **final-step** feature from the hidden state immediately before the last conclusion marker (e.g., phrases such as `therefore`, `so the answer is`, or equivalent conclusion patterns), and
- an **answer-pre** feature from the hidden state immediately before the predicted answer token.

In our implementation, if an answer token can be located, we prefer the last conclusion marker that appears before that answer; otherwise we use the last detected conclusion marker in the response.

A.4.2 Implementation details for learning path-quality directions.

For each phase, we train an independent pairwise logistic ranking model. The input feature is the mean hidden state over the four tokens immediately preceding the extracted boundary, corresponding to the `window4_mean` feature. The preprocessing map ϕ_b is fitted separately for each phase using only training questions. It consists of standardization followed by PCA:

$$\phi_b(h) = P_b \left(\frac{h - \mu_b}{\sigma_b} \right), \tag{15}$$

where μ_b and σ_b are the training-set mean and standard deviation for phase b, and P_b is the PCA projection matrix. We use at most 128 PCA components, with the actual number of components set to

$$d_b = \min(128, N_b - 1, D), \tag{16}$$

where N_b is the number of training states for phase b and D is the original hidden dimension. If PCA is not applicable, ϕ_b reduces to standardization.

For each question and phase, we form all positive–negative pairs from $\mathcal{H}^+_{q,b} \times \mathcal{H}^-_{q,b}$. To control the number of training examples, we keep at most 128 positive–negative pairs per question, sampled uniformly when more are available. In the logistic-regression implementation, we add both the forward difference and the reversed difference:

$$x = \phi_b(h^+) - \phi_b(h^-), \quad y = 1, \qquad x' = \phi_b(h^-) - \phi_b(h^+), \quad y' = 0. \tag{17}$$

This is equivalent to optimizing the ranking objective in Eq. 11.

We use `sklearn.linear_model.LogisticRegression` with `solver=lbfgs`, `class_weight=balanced`, $C = 1.0$, and `max_iter=2000`. All splits are made at the question level: training, validation, and test questions are disjoint, and pairs are constructed only within the corresponding split.

For steering, the learned coefficient $\tilde{w}_b$ is mapped back to the raw hidden-state space. When PCA is used as in Eq. 15, the raw-space direction is

$$w_b = \mathrm{diag}(\sigma_b)^{-1} P_b^\top \tilde{w}_b, \qquad \bar{w}_b = \frac{w_b}{\|w_b\|_2}. \tag{18}$$

The normalized direction $\bar{w}_b$ is used for phase-wise steering.

A.4.3 Details for PAIR Direction Learning and Steering

Feature Preprocessing and Linear Scorer For each phase b, we train an independent linear path-quality scorer using only training questions. The raw feature h is the mean hidden state over the four tokens immediately preceding the extracted reasoning boundary:

$$h = \frac{1}{4} \sum_{r=1}^{4} H^{(\ell)}_{t-r}, \tag{19}$$

where $H^{(\ell)}_{t-r}$ is the token-level hidden state at layer ℓ.

Each phase has its own preprocessing map. We first standardize raw features using the training-set mean and standard deviation:

$$\bar{h} = \frac{h - \mu_b}{\sigma_b}. \tag{20}$$

We then apply PCA when the number of available training states is sufficient:

$$\phi_b(h) = P_b \bar{h}, \tag{21}$$

where P_b is the phase-specific PCA projection matrix. The PCA dimension is set to

$$d_b = \min(128, N_b - 1, D), \tag{22}$$

where N_b is the number of training states at phase b and D is the raw hidden dimension. If PCA is not applicable, ϕ_b reduces to standardization.

The phase-b scorer is

$$\widetilde{s}_b(h) = \tilde{w}_b^\top \phi_b(h) + \beta_b, \tag{23}$$

where $\tilde{w}_b$ and β_b are learned by pairwise logistic regression.

For each question and phase, we form all successful–unsuccessful pairs (h^+, h^-) from the same question. To implement the pairwise objective, we add both the forward and reversed differences:

$$x = \phi_b(h^+) - \phi_b(h^-), \quad y = 1, \qquad x' = \phi_b(h^-) - \phi_b(h^+), \quad y' = 0. \tag{24}$$

We keep at most 128 positive–negative pairs per question and phase, sampled uniformly when more pairs are available.

The logistic-regression model uses `lbfgs`, `class_weight=balanced`, $C = 1.0$, `max_iter=2000`, and random seed 42. All train, validation, and test splits are made at the question level.

Mapping Directions Back to Raw Hidden Space The learned coefficient $\tilde{w}_b$ lies in the preprocessed feature space. For steering, we map it back to the original hidden-state space. When preprocessing consists of standardization followed by PCA, the raw-space direction is

$$v_b = \mathrm{diag}(\sigma_b)^{-1} P_b^\top \tilde{w}_b. \tag{25}$$

We then normalize it:

$$w_b = \frac{v_b}{\|v_b\|_2}. \tag{26}$$

This normalized raw-space direction w_b is used for activation steering.

Thresholded and Clipped Phase-wise Steering For each phase b, we estimate a reference score from successful training trajectories. In our implementation, we use the lower quartile of successful-state scores:

$$\tau_b = \mathrm{Quantile}_{0.25}\left(\left\{\widetilde{s}_b(h) : h \in \mathcal{H}^+_{q,b},\ q \in \mathcal{D}_{\mathrm{train}}\right\}\right). \tag{27}$$

This threshold defines the lower reference range of successful trajectories at phase b.

During generation, let $H_t^{(\ell)}$ be the hidden state at layer ℓ and decoding position t. Let b_t be the phase assigned to the current position by the intervention schedule. The adaptive steering strength is

$$\lambda_t = \min\left\{\lambda_{\max},\ \alpha\left[\tau_{b_t} - \widetilde{s}_{b_t}\left(H_t^{(\ell)}\right)\right]_+\right\}, \tag{28}$$

where $[x]_+ = \max(x, 0)$, α is the global steering scale, and $\lambda_{\max}$ is the maximum allowed intervention magnitude.

The hidden-state update is

$$\widetilde{H}_t^{(\ell)} = H_t^{(\ell)} + \lambda_t w_{b_t}. \tag{29}$$

The modified hidden state is passed to the remaining layers, while all model parameters remain fixed.

Intervention Schedules We evaluate two schedules.

Single-phase steering. For a selected phase b, the intervention is applied only when the current position belongs to that phase:

$$b_t = b \quad \Rightarrow \quad \widetilde{H}_t^{(\ell)} = H_t^{(\ell)} + \lambda_t w_b. \tag{30}$$

No intervention is applied outside the selected phase.

Progressive steering. In progressive steering, the intervention follows the trajectory phase:

$$\widetilde{H}_t^{(\ell)} = H_t^{(\ell)} + \lambda_t w_{b_t}, \qquad b_t \in \{1, \ldots, K\}. \tag{31}$$

Thus, different parts of the generation are steered by the direction learned for their corresponding phase.

Table 4: Steering hyperparameters selected by validation-set sweep. α controls the scale of the adaptive update, and $\lambda_{\max}$ clips the maximum intervention magnitude.

Benchmark	α	$\lambda_{\max}$
GSM8K	5	5
MATH-500	5	5
BBH	2.5	2.5

Steering Hyperparameters For phase-wise steering, we use the adaptive intervention strength defined in Eq. 13. The two main hyperparameters are the global scale α, which controls the size of the score-dependent update, and the clipping threshold $\lambda_{\max}$, which limits the maximum intervention magnitude. We select these values by sweeping a small grid on the validation split and choosing the setting that gives the best validation accuracy while avoiding large correct-to-wrong flip rates. After selection, the same values are fixed and used for all test-set steering evaluations. Table 4 reports the hyperparameters used for the steering experiments in Table 3.

Pointwise Probe Training Details In addition to the final pairwise probe, we train four *single-path* probes. Each of these probes is trained as a binary classifier on **greedy** trajectories, where the label indicates whether the final answer of the trajectory is correct. The learned probe is then evaluated on **sampled** trajectories from the same dataset using within-question ranking metrics and best-of-N selection. The four probes differ only in the hidden-state feature used as input.

Data sources. For each question, we use two kinds of trajectories:

- a single **greedy** trajectory for probe training, and
- a set of **sampled** trajectories (typically $N = 32$) for evaluation.

Greedy trajectories provide one labeled example per question for training. Sampled trajectories provide multiple correct and incorrect candidate paths for the same question, allowing us to evaluate whether a probe score can rank good reasoning paths above bad ones.

Outer train/test split. All splits are defined at the **question level**. Let $\mathcal{Q}$ denote the set of question IDs. We construct a fixed outer train/test split using only greedy-path correctness labels, with a default ratio of $70\%/30\%$. Thus, all greedy and sampled paths from the same question belong to the same outer split, preventing leakage across train and test.

Inner validation for hyperparameter selection. Within the outer training questions, we create an inner validation split used only to select the logistic regression regularization strength. When a dedicated validation split is not available, we create one by splitting the outer training questions again, typically using an inner validation ratio of 0.2. The split is also performed at the question level. If stratified splitting is not possible due to small class counts, we fall back to an unstratified random split.

Feature representation. Each of the four probes takes as input a single hidden-state vector extracted from one location in the trajectory. In all cases, the hidden state is represented as the mean of the **four tokens immediately preceding** the target position. Let $h_{t-3}, h_{t-2}, h_{t-1}, h_t \in \mathbb{R}^d$ denote the hidden states of the four tokens ending at the target token position t. The feature vector is

$$x = \frac{1}{4} \sum_{i=0}^{3} h_{t-i} \in \mathbb{R}^d.$$

The four probes are defined as follows:

1. **Answer-pre probe.** We locate the predicted answer token and use the mean hidden state of the four tokens immediately preceding that answer position.

2. **Relative-position-25% probe.** Let L be the token length of the cleaned response. We compute the target token index as

$$t_{0.25} = \mathrm{round}(0.25 \cdot (L-1)),$$

and use the mean hidden state of the four-token window ending at $t_{0.25}$.

3. **Step-3 probe.** We run the step marker extraction procedure described in Appendix A.4.1. By our convention, the hidden state of step k is taken from the hidden state immediately before the marker of step $k+1$. The step-3 probe therefore uses the four-token mean immediately before the marker of step 4. If a trajectory does not contain such a step boundary, it is skipped for this probe.

4. **80%-step probe.** We first extract the ordered list of reasoning-step features for the trajectory, including the final-step feature defined by the hidden state before the conclusion marker. If there are m such reasoning-step features, we select the step at index

$$j_{0.8} = \mathrm{round}(0.8 \cdot (m-1)),$$

and use that step's four-token mean hidden state as the probe input.

Training labels. For greedy training, each question contributes one trajectory with a binary label

$$y \in \{0, 1\},$$

where $y = 1$ indicates that the final predicted answer is correct and $y = 0$ indicates that it is incorrect.

Preprocessing. For each probe, we collect all training feature vectors from the greedy training split. Let the raw feature matrix be

$$X \in \mathbb{R}^{n \times d}.$$

We first standardize features dimension-wise using a `StandardScaler`:

$$\tilde{X}_{ij} = \frac{X_{ij} - \mu_j}{\sigma_j}.$$

We then fit PCA on the standardized training features and project them to a low-dimensional space. If the requested PCA dimension is k, the actual number of retained components is

$$k' = \min(k, n-1, d).$$

In our experiments the default target dimension is $k = 128$, so typically the probe input after PCA is

$$z \in \mathbb{R}^{128}.$$

Classifier. Each probe is a logistic regression classifier trained on the PCA-transformed features. For a transformed feature vector z, the probe predicts

$$s(z) = w^\top z + b,$$

and the probability of correctness is

$$p(y = 1 \mid z) = \sigma(w^\top z + b),$$

where $\sigma(\cdot)$ is the logistic sigmoid.

We use `sklearn.linear_model.LogisticRegression` with:

- solver: `lbfgs`,
- class weighting: `balanced`,
- maximum iterations: `5000`,
- random seed fixed for reproducibility.

Hyperparameter selection. The main tuned hyperparameter is the inverse regularization strength C of logistic regression. We evaluate a candidate set such as

$$C \in \{0.03, 0.1, 0.3, 1.0, 3.0\}.$$

For each candidate C, we train on the greedy inner-training questions and evaluate on sampled paths from the inner-validation questions. Although the probe is trained on single-path correctness labels, model selection is based on **within-question ranking performance** on sampled validation paths, since ranking is the downstream behavior of interest.

Concretely, for each candidate C, we score sampled validation paths and compute:

- pooled within-question AUC,
- path-level AUROC across sampled paths,
- and optionally best-of-N accuracy.

We choose the candidate with the best validation pooled within-question AUC; ties are broken by path-level AUROC and then by preferring smaller C (stronger regularization).

Refitting. After selecting $C^\star$, we refit the probe on all greedy trajectories from the outer training questions using the same preprocessing pipeline:

1. fit `StandardScaler` on outer-train greedy features,
2. fit PCA on standardized outer-train features,
3. train logistic regression with $C^\star$ on the transformed features.

The fitted scaler, PCA parameters, and logistic regression coefficients are saved for later analysis and reuse.

A.5 Compute Resources

The main computational cost of our experiments comes from trajectory sampling and hidden-state caching, rather than from training or steering. For each model and benchmark, we sample 32 trajectories per question and store the hidden states needed for later phase alignment and scoring. On a single NVIDIA A100 GPU, one full trajectory-sampling run for a model–benchmark setting takes approximately 20 GPU-hours, depending on the average generation length of the benchmark and model. Distilled reasoning models tend to require more time because they generate longer reasoning traces. The total compute scales approximately linearly with the number of evaluated model–benchmark settings and the number of sampled trajectories per question.

After trajectories and hidden states are cached, training the PAIR linear scorers is lightweight. The phase-specific logistic-regression models are trained on cached features and typically require only CPU-level computation or a small amount of CPU/GPU time compared with sampling. Steering also adds little overhead during generation, since it only applies a vector update to the hidden state at selected positions and does not require updating model parameters. The main resource bottlenecks are therefore generation time and storage for cached trajectories, token-level hidden states, and extracted step-level features.

A.6 Supplementary MMLU Candidate-Set Analysis

This appendix-only analysis measures answer availability in a separate subject-balanced subset of 500 official MMLU test questions per model. It does not train or evaluate a PAIR scorer on MMLU. For each question, we retain 32 raw independent draws from Llama-3.1-8B-Instruct or Qwen3-4B, sampled with temperature 1.0 and top-p 0.95 under a zero-shot multiple-choice chain-of-thought prompt; Qwen uses its native thinking mode. The first N draws define each candidate set, with duplicate draws retained. Unparsed outputs count toward N but cast no vote. Self-consistency (SC@N) chooses the most frequent parsed answer, breaking ties by the earliest valid draw; Oracle@N is correct if any of the same N candidates contains the correct answer. Thus Oracle is a candidate-set upper bound, not the accuracy of an implemented selector.

Table 5: Self-consistency and oracle accuracy on the same fixed MMLU candidate sets. Each model uses 500 official-test questions; N is the number of raw draws retained per question. Gap is Oracle minus SC in percentage points.

Model	N	SC@N (%)	Oracle@N (%)	Gap (pp)
Llama-3.1-8B	1	68.2	68.2	0.0
Llama-3.1-8B	4	72.4	85.0	12.6
Llama-3.1-8B	8	75.0	90.6	15.6
Llama-3.1-8B	16	76.4	95.4	19.0
Llama-3.1-8B	32	75.0	97.4	22.4
Qwen3-4B	1	83.2	83.2	0.0
Qwen3-4B	4	84.4	90.8	6.4
Qwen3-4B	8	84.2	91.8	7.6
Qwen3-4B	16	84.6	93.0	8.4
Qwen3-4B	32	83.0	93.8	10.8

At $N = 32$, the Oracle–SC gap is 22.4 percentage points for Llama and 10.8 points for Qwen; 95% question-bootstrap intervals from 2,000 resamples are [19.0, 26.2] and [8.0, 13.8] points, respectively. These gaps quantify selection headroom within this MMLU candidate set; they do not imply that PAIR can close the gap. The multiple-choice prompt and dataset differ from the three main PAIR benchmarks, so these accuracies are not directly comparable to Table 2. The SC curve need not increase monotonically with N: adding draws can change the majority-vote answer.