

Tracing Error Propagation in Reasoning Language Models

Dung Huu Nguyen and Quan Pham and Dat Quoc Nguyen

Qualcomm AI Research*

{huudnguy, ngocquan, datnq}@qti.qualcomm.com

Abstract

Reasoning language models (RLMs) increasingly tackle complex problems through long reasoning traces, but identifying when and where failures arise remains difficult. In this work, we study whether reasoning errors are signaled in model representations before they manifest as failed traces. We first show that step-level errors are locally predictable within a short window of neighboring steps, suggesting that models encode early warning signals. However, trace-level failures often result not from isolated errors, but from persistent ones originating earlier in the trace. To study this, we distinguish *recoverable* from *unrecoverable* traces via counterfactual resampling. We find that unrecoverable traces induce final-step representations that are clearly separable from those of recoverable traces. Moreover, the persistent error can be retrospectively decoded from the final-step representation, indicating that error signals propagate and accumulate through the reasoning trace.

1 Introduction

Chain-of-Thought (CoT) prompting has become a standard technique for eliciting multi-step reasoning in large language models (LLMs), substantially improving performance on tasks requiring compositional and procedural reasoning (Wei et al., 2022; Kojima et al., 2022). The emergence of reasoning language models (RLMs)—trained explicitly to produce long, deliberate reasoning traces before answering—has further extended this paradigm, enabling competitive performance on mathematical, scientific, and code reasoning benchmarks (Grattafiori et al., 2024; Guo et al., 2025; Yang et al., 2025). However, these long traces also introduce a fundamental challenge: errors can arise at any step, and once an early mistake goes uncorrected, subsequent steps may build upon a flawed

prefix, allowing the error to silently propagate until it surfaces as an incorrect final answer. Detecting such failures early is therefore critical, both for reducing wasted computation and for enabling targeted correction, resampling, or self-verification.

Prior work has approached this problem through a range of strategies, including static error taxonomies (Anonymous, 2026; Li et al., 2025), trace-level reliability estimation (Sun et al., 2026; Frasca et al., 2026), and step-level detection methods (Lu et al., 2026a; She et al., 2025). While effective for verification or outcome selection, these methods primarily identify errors at the surface or local step level. They do not distinguish errors that the model can later recover from those that push the trace toward an unrecoverable failure—a distinction central to understanding when intervention is necessary and when continued generation is futile.

To fully diagnose and mitigate reasoning breakdowns, we argue that the analysis must move beyond surface-level text and into the dynamics of the model’s internal representations. Recent work has shown that hidden states encode rich information about model correctness, often before that information surfaces in the generated text (Zhang et al., 2025a; Azaria and Mitchell, 2023; Sun et al., 2026). Building on these findings, we investigate the latent mechanisms of error propagation by directly probing the internal representations of RLMs. Specifically, we distinguish three classes of failure: *step-level errors* that occur at individual reasoning steps, *persistent errors* that originate earlier in the trace and continue to influence subsequent steps, and *unrecoverable failures*—traces from which no continuation can recover the correct answer, identified through counterfactual resampling. Using these definitions, we probe hidden representations to test whether error signals are detectable locally, distinguishable at the trace level, and traceable to their origin within the trace.

Our investigation yields three primary findings

*Qualcomm Vietnam Company Limited. Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

about the structure of reasoning failures in RLMs:

- **Local Predictability.** Step-level errors are locally predictable within a short window of neighboring steps, indicating that models internally encode shared error signals around the onset of a mistake. Notably, predicting an error at the current step using only the surrounding context—without access to the step’s own representation—still yields strong performance, suggesting that error signals diffuse beyond the step at which they originate.
- **Trace-Level Footprints.** Beyond local detectability, we find that unrecoverable traces leave distinct, separable footprints in the model’s final-step representation, clearly distinguishable from those of recoverable traces. This indicates that representations encode not only that a trace contains an error, but also whether the error has compromised the entire reasoning trajectory.
- **Retrospective Tracing.** The final-step representation retains information about the earlier persistent error, even after many subsequent steps have been generated. Using a probe that compares the final representation to candidate step representations, the source of a persistent error can be retrospectively decoded from the final step representation.

Together, these findings show that reasoning failures leave detectable signatures in RLM hidden representations. We demonstrate one such application: a probe-guided prefix resampling strategy that uses error localization to selectively regenerate only the corrupted portion of a trace, achieving accuracy comparable to budget-matched baselines at substantially lower computational cost. More broadly, our work points toward error-aware decoding strategies and the development of reliable, self-correcting reasoning systems.

2 Related Work

Reasoning Failures and Mitigation A growing line of work studies reasoning hallucinations and failure modes in long CoT reasoning. Unlike factual hallucinations, reasoning failures often arise from errors in the intermediate reasoning process, such as unsupported assumptions, invalid deductions, or computational mistakes that propagate to later steps (Song et al., 2026; Lu et al., 2026a; Sun et al., 2026). Prior work has largely studied such failures through taxonomies. For example, Anonymus (2026) distinguish premise, operation, logic,

and conclusion errors, while Li et al. (2025) separate intrinsic errors, such as logical inconsistency, from extrinsic errors, such as calculation errors. These taxonomies clarify failure types, but do not address whether an error is later corrected, persists in the trace, or becomes unrecoverable.

Existing methods to detect and mitigate reasoning failures typically rely on internal signals (Srey et al., 2025), external verification (Li et al., 2025), or sampling-based strategies (Deng et al., 2026), with process reward models that evaluate intermediate steps for rejection sampling or best-of-N selection (Zhang et al., 2025b). Other approaches probe internal dynamics: Sun et al. (2026) derive a Reasoning Score from layer-wise divergence to detect hallucinations and provide training rewards; Lu et al. (2026a) model hallucination as a temporal state by combining step- and prefix-level evidence; and Frasca et al. (2026) formulate detection as graph classification over attention graphs. While these methods provide useful trace-level estimates, they do not clearly distinguish errors that the model can recover from and errors that lead to eventual failure. We address this gap by separating recoverable traces from unrecoverable traces under resampling, then examining how different error types leave different signatures in the step representation.

LLM Internal Signals and Probing Prior work has shown that LLM internal signals—including hidden states (Zhang et al., 2025a; Azaria and Mitchell, 2023), attention scores (Lu et al., 2026b), layer-wise divergence (Mir, 2025), and neuron activations (Ji et al., 2024)—carry useful information about response reliability. Such signals are particularly useful for reasoning models, where long multi-step traces make external supervision difficult. Recent studies use internal representations to diagnose reasoning failures: hidden-state probes predict correctness (Zhang et al., 2025a), hallucinating models show distinctive attention patterns (Tang et al., 2026), incorrect tokens exhibit elevated cross-layer logit fluctuations (Sun et al., 2026), and flawed CoT traces differ structurally from correct ones (Zhao et al., 2026). Internal states have also been used to study broader reasoning behavior, such as early answer encoding (Zhang et al., 2025a; Cox et al., 2026) and adaptive self-reflection (Chen et al., 2025).

The mentioned findings have motivated our use of hidden representations to analyze and trace the error propagation process in reasoning, through three unexplored research questions in Section 4.

3 Modeling Error Propagation

We formalize the reasoning process and the error types we investigate. Departing from prior work that examines isolated step-level mistakes, we introduce the first analytical framework for systematically characterizing how local errors emerge, persist, and propagate within a reasoning trace. Specifically, we (i) define trace representations (Section 3.1), (ii) operationalize persistent errors via the earliest observed error (Section 3.2), and (iii) introduce a stricter, resampling-based notion of unrecoverable errors (Section 3.3).

3.1 Reasoning Traces and Representations

We consider a reasoning task in which a RLM model produces a final answer y through a sequence of intermediate reasoning steps, conditioned on an input query q . We represent a reasoning trace as:

$$s_{1:T} = (s_1, s_2, \dots, s_T)$$

where T is the total number of steps and each s_t denotes the text span of the t -th reasoning step (i.e., a sequence of tokens generated by the model).

For each step s_t , we define its internal representation $\mathbf{h}_t \in \mathbb{R}^d$ as the average-pooled vector of the model’s last-layer hidden states over all tokens in s_t . The representations $\mathbf{h}_t$ are expected to encode latent signals about the reasoning trace, including the presence or onset of errors. However, since such errors are most naturally identified from the generated text, we rely on surface-level annotations of each step s_t as observable indicators of reasoning errors. These annotations serve as supervision signals, which we use to probe whether $\mathbf{h}_t$ encodes the corresponding error information.

3.2 From Step Errors to Persistent Errors

To study error propagation, we must distinguish between transient, surface-level step errors and persistent errors that propagate through and corrupt the subsequent reasoning trace.

Lu et al. (2026a) introduce two complementary step-level labels that distinguish localized errors from cumulative corruption of the reasoning trace:

- *Step-level Error*: Let $e_t \in \{0, 1\}$ denote whether step s_t contains a localized reasoning error (e.g. a logical fallacy or calculation error), with $e_t = 1$ indicating the presence of such an error in s_t .
- *Cumulative Error*: Let $c_t \in \{0, 1\}$ denote whether the reasoning prefix $s_{1:t}$ is corrupted,

where $c_t = 1$ denotes that the prefix contains errors that directly compromise the correctness of the reasoning trace.¹

Here, e_t and c_t serve as intermediate signals for defining our main object of study: the earliest persistent error in a trace. In many incorrect traces, the model makes an error and fails to recover, so all subsequent steps build upon the corrupted prefix.

We define the **persistent error** step (i.e. the source of error propagation), denoted t_{pers}^* , as the first step beyond which the reasoning trace remains corrupted ($c_t = 1$) through the end of the trace:

$$t_{\text{pers}}^* = \min \{t \in \{1, \dots, T\} \mid \forall t' \geq t, c_{t'} = 1\}$$

This formulation shifts the analysis from isolated errors to the precise step at which error propagates without recovery in the observed trace.

3.3 Unrecoverable Errors under Resampling

While t_{pers}^* captures persistence within a single observed trace, it does not necessarily reflect an inherent inability of the model to self-correct. Recent works (Macar et al., 2026; Hu et al., 2026) suggest that understanding reasoning failures in LLMs requires analyzing the distribution of potential continuations rather than individual generations.

To rigorously test error propagation in a counterfactual manner, we introduce the notion of an **unrecoverable error under resampling**, defined as the earliest step at which the probability of sampling a correct continuation from the corrupted prefix falls below a sufficiently small threshold.

Formally, we define the recoverability of the t -th step, denoted ρ_t , as the true probability that the model ultimately produces the correct final answer y^* , conditioned on the input query prompt q and the generated prefix $s_{1:t}$:

$$\rho_t = \Pr_{s_{t+1:T}, y \sim \pi_\theta(\cdot | q, s_{1:t})} (y = y^*)$$

where the suffix $s_{t+1:T}$ and the final answer y are sampled from the model’s generation policy π_θ .

We also define an ϵ -unrecoverable point as the first step at which the true recovery probability falls below a predefined threshold ϵ :

$$t_{\text{unre}}^* = \min \{t \in \{1, \dots, T\} \mid \rho_t \leq \epsilon\}$$

Here, $\epsilon = 0$ represents an idealized “point of no recovery”, while in practice we operate with $\epsilon > 0$.

¹Minor issues such as grammatical errors or typos are excluded, as they do not affect the validity of the reasoning trace.

Because the exact computation of ρ_t is intractable, we estimate it using finite resampling with a fixed budget of R samples (i.e., R independent resampling runs), resulting in the empirical recoverability $\hat{\rho}_t$:

$$\hat{\rho}_t = \frac{1}{R} \sum_{r=1}^R \mathbb{I}(y^{(r)} = y^*)$$

Based on this estimate, we define the empirical unrecoverable point as:

$$\hat{t}_{\text{unre}}^* = \min \{t \in \{1, \dots, T\} \mid \hat{\rho}_t \leq \epsilon\}$$

Under the Clopper–Pearson interval, when no successful samples are observed (i.e. $\hat{\rho}_t = 0$), the $1 - \alpha$ upper confidence bound on the true recovery probability ρ_t is $1 - \alpha^{1/R}$ (α denotes the significant level).

Importantly, the observed persistent error index t_{pers}^* is structurally related to the estimated unrecoverable boundary $\hat{t}_{\text{unre}}^*$. By analyzing alternative continuations, we can further categorize failure modes into three distinct cases:

- **Immediate Unrecoverability** ($\hat{t}_{\text{unre}}^* = t_{\text{pers}}^*$): All sampled continuations from the initial error fail, indicating that the error itself marks the point of no recovery.
- **Delayed Unrecoverability** ($\hat{t}_{\text{unre}}^* > t_{\text{pers}}^*$): The model can still recover from the initial error under some alternative continuations. However, along the *observed* trace, it later transitions into an irreversible state. In this case, $\hat{t}_{\text{unre}}^* = t_{\text{pers}}^* + k$, where $0 < k \leq T - t_{\text{pers}}^*$.
- **Persistent Error with Recoverable Trace** (t_{pers}^* exists, but $\hat{t}_{\text{unre}}^*$ is undefined): The observed trace ultimately produces an incorrect final answer, but the empirical recovery probability never falls below ϵ . Thus, the model sampled an unfavorable continuation along the observed trace, even though viable recovery paths remained available. We classify such traces as *recoverable* ones and discuss them in Section 6.

Key Insight of Delayed Unrecoverability:

A persistent error is not always immediately unrecoverable. The model may still recover from the initial error under continuations, but later steps can build on the corrupted prefix and make recovery increasingly unlikely. We refer to this transition as delayed unrecoverability.

Under our formulation, t_{pers}^* identifies the initial *source* of the error cascade, whereas $\hat{t}_{\text{unre}}^*$ marks the point where the model can no longer recover. This distinction allows us to analyze how error signals propagate across the reasoning trace, from the persistent error to the point of unrecoverability.

- We define **trace-level errors** as cases where the reasoning trace results in an incorrect final answer and becomes empirically unrecoverable.
- We introduce **retrospective tracing**, a novel diagnostic analysis that examines whether the final-step representation from a trace-level failure retains information useful for identifying a target error step within the trace.

3.4 Probe Training

Given a reasoning trace, we assume access to both step-level and trace-level error labels. In particular, each step s_t is annotated with a binary label $e_t \in \{0, 1\}$, where $e_t = 1$ indicates that the step is erroneous. The full trace is annotated with a binary label $\tau \in \{0, 1\}$, where $\tau = 1$ indicates that the trace yields an incorrect final answer and becomes empirically unrecoverable, and $\tau = 0$ otherwise.

To study errors at both levels, we train two separate Multilayer Perceptron (MLP) probes: a step-level probe and a trace-level probe. The step-level probe predicts e_t from the step representation $\mathbf{h}_t$, while the trace-level probe predicts τ from the final-step representation $\mathbf{h}_T$. Both MLP probes use the same architecture and are trained using binary cross-entropy loss. The probes’ architecture and training setup are described in Appendix A.

Regarding retrospective tracing, for each candidate step s_t , we construct the concatenated representation $[\mathbf{h}_t \oplus \mathbf{h}_T]$ and feed it to a separate MLP probe trained to predict whether t matches the target persistent error index t_{pers}^* .

4 General Experimental Setup

4.1 Experimental Research Questions

Our investigation is organized around three research questions that, to our knowledge, have not been systematically explored before:

- **Local Predictability (Section 5)**: How early and how far around a failure point can error signals be detected? We test whether these signals are confined to the failure step or propagate to neighboring steps.

	DSR8B	Qwe7B	Lla8B
# Total Traces	2808	2841	2514
– Train	1404	1422	1319
– Val	842	854	659
– Test	562	565	536
# Total steps	89,918	34,611	43,007
Steps/trace (mean $\pm$ std)	32.0 $\pm$ 25.9	17.8 $\pm$ 5.6	23.3 $\pm$ 10.4
Trace length (chars) (mean $\pm$ std)	3571 $\pm$ 1696	1695 $\pm$ 517	3556 $\pm$ 1698
# Traces with t_{pers}^*	1330	1939	1843
t_{pers}^* position (mean $\pm$ std)	17.2 $\pm$ 18.7	9.0 $\pm$ 3.6	11.5 $\pm$ 6.5

Table 1: CHAINED dataset statistics across DeepSeek-R1-Distill-Llama-8B (DSR8B), Qwen2.5-7B-Instruct (Qwe7B) and Llama-3.1-8B-Instruct (Lla8B).

- **Trace-Level Footprints (Section 6):** Do unrecoverable traces leave stronger global signals than recoverable traces? We examine whether unrecoverable traces are more distinguishable in hidden-state space.
- **Retrospective Tracing (Section 7):** Can the final-step representation help identify a target error step? We evaluate whether signals associated with t_{pers}^* remain decodable from the final-step representation together with candidate-step representations.

4.2 Experimental Datasets

We ground our analysis in CHAINED (Lu et al., 2026a), a publicly available CC BY 4.0 dataset² of about 8.5K reasoning traces annotated with both step-level errors (e_t) and cumulative errors (c_t), derived from MuSiQue (Trivedi et al., 2022) and BIG-Bench Hard (Suzgun et al., 2023). This **base** dataset spans generations from three distinct models: Llama-3.1-8B-Instruct (Lla8B) (Grattafiori et al., 2024), Qwen2.5-7B-Instruct (Qwe7B) (Yang et al., 2024), and DeepSeek-R1-Distill-Llama-8B (DSR8B) (Guo et al., 2025), and covers a range of challenging reasoning tasks, including multi-hop question answering and complex logical reasoning. Dataset statistics are reported in Table 1.

CHAINED supports our local step-level error analysis (Section 5) and our t_{pers}^* -based source-error tracing analysis (Section 7). For the trace-level comparison in Section 6, however, CHAINED is insufficient: it provides no labels for $\hat{t}_{\text{unre}}^*$, and its traces could not be inherently classified as recoverable or unrecoverable. To address this, we generate 16 additional rollouts per CHAINED prompt and apply a binary-search trace slicing procedure inspired by OmegaPRM (Luo et al., 2024) to assign each trace a *correct*, *recoverable* or *unrecoverable*

label. We provide full details of the labeling process in Appendix B.

4.3 Evaluation Metrics

Step-Level and Trace-Level Detection. For both local step-level error prediction and trace-level classification, we evaluate probe performance using standard metrics: AUC and F1.

Target Failure Step Tracing. We evaluate the probe’s ability to retrospectively rank the target failure step against other candidate steps using AUC, Recall@K, MRR, and **b-MRR**—a bounded variant we introduce. Since errors tend to cascade through subsequent reasoning, steps after the target failure step often share similar error signals, confounding standard ranking evaluation. To address this, **b-MRR** restricts the candidate pool to steps that strictly *precede* the target failure step, isolating how its representation differs from those of earlier, uncorrupted steps.

5 Locality of Step-Error Signal

Setup. To examine whether the step-level error signal leaks into neighboring steps, we evaluate local error prediction under two settings: (1) *Offset-based Probing*: For each offset $k \in [-5, 5]$, we train a probe that takes the representation $\mathbf{h}_t$ of step s_t as input and predicts the label associated with step s_{t+k} . For example, $k < 0$ corresponds to using the current step’s representation to predict the label of an earlier step, while $k > 0$ corresponds to predicting the label of a later step. (2) *Context-only Probing*: We ensemble representations from the earlier and later steps — explicitly excluding the current step — to determine whether the surrounding context alone provides enough signal to predict the current step’s error.

Locality of Signals. Table 2 reports the results for the Offset-based Probing setting. Error signals peak precisely at the error step ($k = 0$) and decline gradually as $|k|$ increases, yet remain strongly discriminative across all offsets. Notably, we observe a directional asymmetry in the ranking metric AUC: $\mathbf{h}_t$ better discriminates errors at *earlier* steps ($k < 0$) than at *later* steps ($k > 0$). Comparing $k = -5$ to $k = +5$, AUC is consistently higher across all three models, suggesting that $\mathbf{h}_t$ retains a stronger signal of past errors than it anticipates future ones. Interestingly, this asymmetry reverses under the classification metric F1: predicting the labels of *later* steps ($k = +5$) is more accurate than

²<https://github.com/snowcatsmoking/StreamingHallucination>

	DSR8B		Lla8B		Qwe7B	
k	AUC	F1	AUC	F1	AUC	F1
-5	.873	.566	.790	.598	.734	.563
-4	.873	.574	.793	.606	.743	.586
-3	.881	.590	.795	.619	.749	.604
-2	.890	.605	.802	.635	.756	.626
-1	.899	.622	.811	.652	.771	.649
0	.922	.654	.851	.703	.814	.684
+1	.883	.632	.778	.645	.755	.654
+2	.872	.630	.751	.634	.721	.647
+3	.865	.612	.733	.631	.700	.648
+4	.855	.592	.721	.632	.679	.650
+5	.853	.578	.704	.632	.656	.655

Table 2: Offset-based probing results—using the current step’s representation $\mathbf{h}_t$ to predict the label of step s_{t+k} . Here, DSR8B, Lla8B, and Qwe7B abbreviate DeepSeek-R1-Distill-Llama-8B, Llama-3.1-8B-Instruct, and Qwen2.5-7B-Instruct, respectively.

Probing Features	DSR8B	Lla8B	Qwe7B	Avg. Δ
$\mathbf{h}_t$ only	.922	.851	.814	–
pool($\mathbf{H}_{t,1}$)	.886	.813	.776	-.037
concat($\mathbf{H}_{t,1}$)	.923	.836	.797	-.011
pool($\mathbf{H}_{t,1}^{-t}$)	.876	.793	.758	-.053
concat($\mathbf{H}_{t,1}^{-t}$)	.903	.802	.766	-.039
pool($\mathbf{H}_{t,2}$)	.878	.797	.754	-.053
concat($\mathbf{H}_{t,2}$)	.924	.830	.788	-.015
pool($\mathbf{H}_{t,2}^{-t}$)	.872	.783	.741	-.064
concat($\mathbf{H}_{t,2}^{-t}$)	.910	.799	.759	-.040
pool($\mathbf{H}_{t,3}$)	.870	.785	.732	-.067
concat($\mathbf{H}_{t,3}$)	.920	.826	.771	-.023
pool($\mathbf{H}_{t,3}^{-t}$)	.865	.774	.721	-.076
concat($\mathbf{H}_{t,3}^{-t}$)	.906	.797	.746	-.047

Table 3: AUC results for predicting the current step’s label w.r.t. Full-window Ensemble (including $\mathbf{h}_t$) vs. Context-only Ensemble (excluding $\mathbf{h}_t$). Here, $\mathbf{H}_{t,w} = \{\mathbf{h}_{t+k} \mid -w \leq k \leq w\}$ denote the full window of $2w + 1$ step representations centered at $\mathbf{h}_t$, and $\mathbf{H}_{t,w}^{-t} = \mathbf{H}_{t,w} \setminus \{\mathbf{h}_t\}$ denote the corresponding context-only window. pool() averages the step representations; while concat() concatenates them. “Avg. Δ ” reports the average AUC change relative to the baseline of using $\mathbf{h}_t$ alone as input, averaged across all three models. Results with $w \in \{4, 5\}$ are presented in Appendix C.

predicting the labels of *earlier* steps ($k = -5$). A likely explanation is that later-step representations carry propagated error signals, biasing the probe to over-predict errors on subsequent steps and degrading exact localization. Together, these results confirm that error signals spread across both earlier and later neighboring steps.

Complementary Error Signals. Table 3 reports AUC results for predicting the current step’s label from an ensemble of neighboring step representations. Integrating neighboring representations—whether by concatenation or average pooling—does not strictly improve over the $\mathbf{h}_t$ -only baseline. Among the two variants, concatenation consistently outperforms average pooling, suggesting that preserving the structural relationship between steps carries useful signal. For example, the concatenation ensemble concat($\mathbf{H}_{t,1}$) shows only a marginal drop (Avg. $\Delta = -0.011$), while average pooling pool($\mathbf{H}_{t,1}$) degrades more noticeably (Avg. $\Delta = -0.037$). On DSR8B, several concatenation variants even match or slightly exceed the $\mathbf{h}_t$ -only baseline of 0.922—e.g., 0.923, 0.924, and 0.920 for window radii $w = 1, 2, 3$, respectively.

Importantly, predicting errors using *only* contextual representations while entirely excluding the current step ($\mathbf{H}_{t,w}^{-t}$) still yields strong performance. On DSR8B, concat($\mathbf{H}_{t,1}^{-t}$) achieves an AUC of 0.903, and concat($\mathbf{H}_{t,2}^{-t}$) reaches 0.910. This strong performance from purely contextual representations confirms that a *shared error signal* bleeds into the surrounding steps.

6 Trace-level Signals are Distinguishable Across Error Types

Trace-level Classification Setup. As described in Section 4.2, we construct trace-level labels by generating additional reasoning traces from the same CHAINED prompts and estimating prefix-level recoverability through resampling. This yields three categories: *Correct*, *Recoverable*, and *Unrecoverable*. Beyond the three base models in CHAINED, we apply the same generation and labeling pipeline to a set of newer models from the Qwen3 family (Yang et al., 2025)—Qwen3-4B, Qwen3-8B, and Qwen3-14B—to verify that the observed trace-level patterns are not specific to the models used in the CHAINED paper but generalize across recent architectures. Full details of the data generation process are provided in Appendix B.³

As described in Section 3.4, the trace-level probes take the final-step representation $\mathbf{h}_T$ as input, with a separate probe trained for each pairwise classification among the three categories *Correct*, *Recoverable*, and *Unrecoverable*: *Correct* $\leftrightarrow$

³Note that traces generated by the Qwen3 models do not contain step-level annotations; these models are therefore used only for the trace-level classification experiments.

Pair	Model	AUC	F1
Recover. $\leftrightarrow$ Unrecover.	DSR8B	.877	.743
	Qwe7B	.768	.710
	Lla8B	.825	.756
	Qwen3-4B	.905	.807
	Qwen3-8B	.893	.796
	Qwen3-14B	.853	.770
Correct $\leftrightarrow$ Recover.	DSR8B	.971	.907
	Qwe7B	.977	.914
	Lla8B	.908	.857
	Qwen3-4B	.949	.875
	Qwen3-8B	.965	.905
	Qwen3-14B	.965	.907
Correct $\leftrightarrow$ Unrecover.	DSR8B	.965	.906
	Qwe7B	.940	.851
	Lla8B	.947	.850
	Qwen3-4B	.961	.893
	Qwen3-8B	.967	.898
	Qwen3-14B	.963	.895

Table 4: Trace-level classification performance across different models. “Recover.” and “Unrecover.” abbreviate recoverable and unrecoverable, respectively. Here, DSR8B, Lla8B, and Qwe7B denote DeepSeek-R1-Distill-Llama-8B, Llama-3.1-8B-Instruct, and Qwen2.5-7B-Instruct, respectively. Appendix C provides additional results for length-controlled settings and validates the robustness of our recoverability threshold.

Recoverable, Correct $\leftrightarrow$ Unrecoverable, and Recoverable $\leftrightarrow$ Unrecoverable.

Recoverable and Unrecoverable Traces are Separable. Table 4 reports pairwise classification results among *Correct*, *Recoverable*, and *Unrecoverable* traces, using only the final-step representation as input. The high AUCs on *Correct $\leftrightarrow$ Recoverable* and *Correct $\leftrightarrow$ Unrecoverable* show that the final-step representation reliably separates correct traces from incorrect ones across most models. More importantly, *Recoverable* and *Unrecoverable* failures are themselves separable: the *Recoverable $\leftrightarrow$ Unrecoverable* probe achieves AUCs of 0.877 on DSR8B, 0.825 on Lla8B, and up to 0.905 on Qwen3-4B. This suggests that unrecoverable failures leave a distinct fingerprint in the final-step representation, even though both trace types produce incorrect answers.

When does a Persistent Error become Unrecoverable? Figure 1 measures the relative delay between the initial error index t_{pers}^* and the empirical unrecoverable point $\hat{t}_{\text{unre}}^*$. At $\epsilon = 0.0$, a prefix is considered unrecoverable only if *none* of the sampled continuations reach the correct answer. Under this strict definition, $\hat{t}_{\text{unre}}^*$ tends to occur well af-

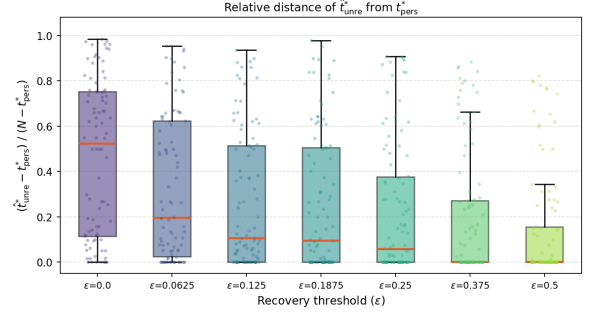


Figure 1: **Relative delay of unrecoverability across varying recovery thresholds ϵ .** The y-axis measures how far the empirical unrecoverable point $\hat{t}_{\text{unre}}^*$ occurs after the persistent error t_{pers}^* , normalized by the remaining trace length: $(\hat{t}_{\text{unre}}^* - t_{\text{pers}}^*) / (N - t_{\text{pers}}^*)$. Larger ϵ values treat low-probability recovery as unrecoverable, moving $\hat{t}_{\text{unre}}^*$ closer to t_{pers}^* . Results are shown for DSR8B.

ter t_{pers}^* , with a median relative delay of about 0.5. As ϵ increases (i.e., the recoverability threshold becomes less strict), $\hat{t}_{\text{unre}}^*$ shifts closer to t_{pers}^* . This suggests that although persistent errors may not immediately eliminate all successful continuations, they often reduce the probability of recovery soon after they appear.

7 Final States Retain the Source of Error Propagation

7.1 Retrospective Tracing

Setup. Beyond detecting that a trace has become corrupted, we ask whether its final representation still retains enough information to reveal where the persistent error began. We apply the retrospective tracing probe (Section 3.4) to evaluate candidate steps at varying offsets $k \in [-3, 3]$ relative to the persistent error index t_{pers}^* , with $k = 0$ corresponding to the error onset. Specifically, for each offset k , we train a **separate binary probe** over candidate steps t . The probe takes $h_T \parallel h_t$ as input and assigns a positive label only when $t = t_{\text{pers}}^* + k$; other candidate steps in the same trace are treated as negatives, where $\parallel$ denotes concatenation. For reference baseline, we include a random approach that assigns uniform scores to all candidate steps within a trace. Performance is measured using the source tracing metrics defined in Section 4.3.

Results. Table 5 demonstrates a *surprising property* of error propagation: despite the accumulation of subsequent steps, the final hidden state retains a highly localized signal of the initial error occurrence. Remarkably, source tracing perfor-

Model	k	Ranking		Recall			AUC
		MRR	bMRR	@1	@3	@5	
DSR8B	-3	.171	.613	1.7	.167	.300	.775
	-2	.187	.616	5.0	.183	.350	.798
	-1	.280	.626	10.0	.317	.417	.793
	0	.450	.787	33.3	.500	.652	.896
	+1	.205	.762	11.9	.214	.381	.853
	+2	.182	.639	5.0	.150	.300	.832
	+3	.203	.676	7.9	.158	.342	.823
	Rand.	.190	.300	6.5	.173	.288	—
Qwe7B	-3	.135	.772	1.9	.069	.144	.782
	-2	.146	.753	1.9	.089	.175	.821
	-1	.179	.770	3.1	.144	.247	.858
	0	.350	.885	14.7	.356	.551	.937
	+1	.238	.852	8.0	.252	.415	.913
	+2	.233	.840	7.4	.234	.395	.879
	+3	.210	.822	6.4	.280	.360	.882
	Rand.	.198	.304	5.8	.174	.290	—
Lla8B	-3	.142	.683	2.8	.102	.152	.752
	-2	.155	.731	3.4	.118	.202	.789
	-1	.152	.731	2.5	.109	.230	.814
	0	.369	.791	14.3	.395	.531	.912
	+1	.235	.768	7.3	.188	.300	.877
	+2	.202	.706	7.7	.185	.338	.862
	+3	.227	.662	6.5	.229	.345	.830
	Rand.	.175	.283	5.0	.149	.248	—

Table 5: Retrospective tracing results across offsets k . “Rand.” denotes a random baseline that assigns uniform scores to all candidate steps within a given trace. We provide additional results using the representational difference features as input in Appendix C.

mance peaks precisely at $k = 0$. At this peak, the probe achieves notably high bMRR scores (0.885 on Qwe7B, 0.791 on Lla8B, and 0.787 on DSR8B), outperforming not only random baselines (≈ 0.300) but also all neighboring steps with a strong margin. AUC scores similarly peak at $k = 0$ across all three models. These results indicate that a trained probe can effectively trace back through the reasoning steps to identify the error source after generation. Furthermore, we observe a temporal asymmetry around the peak: candidate steps *after* the persistent error ($k > 0$) yield stronger signals than candidate steps *before* it ($k < 0$). Representations of steps following the persistent error become contaminated by it, carrying heavier residual error signals. Conversely, while candidate steps preceding the persistent error ($k < 0$) yield weaker signals, they remain predictive, likely reflecting the local error footprint observed earlier (Section 5). See qualitative examples in Appendix D.

Method	Tokens	DSR8B	Qwe7B	Lla8B
Greedy	1.0×	65.43	47.68	16.25
SC	5.0×	76.58	54.93	22.08
BoN*	5.0×	79.12	57.17	23.15
Weighted SC*	5.0×	78.84	59.10	23.88
Prefix Resampling*	—	78.16 (2.3×	57.08 (2.8×	21.52 (1.5×

Table 6: Accuracy and token consumption of resampling strategies including both major-voting and verifier-based methods. Token usage is reported relative to the single-sample *Greedy* baseline. * denotes methods involving probe. DSR8B, Lla8B, and Qwe7B denote DeepSeek-R1-Distill-Llama-8B, Llama-3.1-8B-Instruct, and Qwen2.5-7B-Instruct, respectively.

7.2 Error-aware Prefix Resampling

Setup. To evaluate whether our probes can improve inference-time sampling efficiency, we compare several decoding strategies under a budget of 5 sampled trajectories. In addition to *Greedy* decoding and *Self-Consistency* (SC) (Wang et al., 2023), we consider three probe-guided methods:⁴

- *Best-of-N* (BoN) (Brown et al., 2024): samples N independent traces and selects the one with the lowest trace-level probe score (i.e., the least likely to contain errors).
- *Weighted SC*: adapts SC by weighting each candidate’s vote inversely by the trace-level probe’s score, giving more influence to traces predicted as less erroneous.
- *Prefix Resampling*: when the trace-level probe flags a generated trace as erroneous, the retrospective probe identifies the predicted persistent error step. We then truncate and resample from the prefix immediately before this step, repeating until the new trace passes the trace-level check or the sampling budget is exhausted.

Results. As shown in Table 6, the verifier-based methods *BoN* and *Weighted SC* consistently outperform standard *SC*, demonstrating that our trace-level probe provides a stronger correctness signal than simple majority voting. *Prefix Resampling* achieves comparable accuracy at notably lower cost: it reaches 78.16% on DSR8B using only 2.3× tokens, with similar efficiency gains on Qwe7B (2.8×) and Lla8B (1.5×). These results highlight the practical value of error localization for efficiently recovering corrupted traces.

⁴All baseline details are presented in Appendix E.

8 Conclusion

In this work, we investigated the dynamics of error propagation in reasoning language models (RLMs) by distinguishing between localized step errors, persistent errors, and unrecoverable cascading failures. Through probing internal representations, we established three properties of these dynamics: (1) step-level errors are locally predictable within a short window context; (2) unrecoverable failures leave distinct, trace-level footprints, that separate them from recoverable ones; and (3) the source of an error cascade can be retrospectively decoded from the final-step representation. Together, these findings provide empirical evidence that error signals propagate and accumulate over time, opening promising directions for error-aware resampling and the development of reliable, self-correcting reasoning architectures.

Limitations

While our framework offers structural insights into error propagation, its scope is subject to several practical considerations. Our empirical evaluations cover six open-weight models across three architectures up to 14B parameters on multi-hop QA and logical reasoning. While these benchmarks capture structured multi-step reasoning, further investigation is needed to verify whether the observed representational dynamics generalize to broader task domains and frontier-scale architectures. Furthermore, our step-level error probing and tracing analyses rely on localized state annotations and internal hidden representations. Consequently, while our trace-level recoverability analysis is annotation-free (via counterfactual resampling), extending internal diagnostic probes to closed-source, API-only models or open-ended generation tasks remains non-trivial without weak supervision or external proxy estimators.

Methodologically, our recoverability metrics are estimated via finite counterfactual sampling under practical computational budgets. While linear probing serves as an effective diagnostic tool to demonstrate that error signals are decodable from hidden states, fully uncovering the underlying circuit-level mechanisms presents an exciting avenue for future interpretability research. Finally, we demonstrate the downstream utility of internal error signals via Prefix Resampling as a proof-of-concept for targeted backtracking. Developing fully automated, general-purpose mitigation policies, such as inte-

gration into process reward models or dynamic self-correction pipelines, remains an active direction for future work.

Ethics Statement

This work aims to understand error propagation in reasoning language models, contributing to more reliable AI systems. We rely solely on the publicly available dataset and open-weight models — free of PII and sensitive human-subject data — and foresee no malicious dual-use. Although generating reasoning traces requires compute, our analyses use lightweight MLP probes, keeping the overall energy footprint low.

References

- Anonymous. 2026. Mirages of Logic: A Survey of Chain-of-Thought Reasoning Hallucinations. In *Submitted to ACL Rolling Review - January 2026*. Under review.
- Amos Azaria and Tom Mitchell. 2023. The Internal State of an LLM Knows When It’s Lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 967–976.
- Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. 2024. [Large language monkeys: Scaling inference compute with repeated sampling](#). *Preprint*, arXiv:2407.21787.
- Zigeng Chen, Xinyin Ma, Gongfan Fang, Ruonan Yu, and Xinchao Wang. 2025. VeriThinker: Learning to Verify Makes Reasoning Model Efficient. In *Proceedings of the 39th Annual Conference on Neural Information Processing Systems*.
- Kyle Cox, Darius Kianersi, and Adrià Garriga-Alonso. 2026. [Post-hoc reasoning in chain of thought: Decoding and steering pre-committed answers](#). In *Mechanistic Interpretability Workshop at ICML 2026*. ArXiv:2603.01437.
- Yongxin Deng, Zhen Fang, Sharon Li, and Ling Chen. 2026. Beyond In-Domain Detection: SpikeScore for Cross-Domain Hallucination Detection. In *Proceedings of the 14th International Conference on Learning Representations*.
- Fabrizio Frasca, Guy Bar-Shalom, Yftah Ziser, and Haggai Maron. 2026. Neural Message-Passing on Attention Graphs for Hallucination Detection. In *Proceedings of the 14th International Conference on Learning Representations*.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The Llama 3 Herd of Models. *arXiv preprint*, arXiv:2407.21783.

- Daya Guo, Dejian Yang, et al. 2025. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638.
- Zhiyuan Hu, Yucheng Wang, Yufei He, Jiaying Wu, Yilun Zhao, See-Kiong Ng, Cynthia Breazeal, Anh Tuan Luu, Hae Won Park, and Bryan Hooi. 2026. [Rewarding the rare: Uniqueness-aware RL for creative problem solving in LLMs](#). In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 39765–39790, San Diego, California, United States. Association for Computational Linguistics.
- Ziwei Ji, Delong Chen, Etsuko Ishii, Samuel Cahyawijaya, Yejin Bang, Bryan Wilie, and Pascale Fung. 2024. LLM internal states reveal hallucination risk faced with a query. In *BlackboxNLP 2024*.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large Language Models are Zero-Shot Reasoners. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*.
- Ruosen Li, Ziming Luo, and Xinya Du. 2025. FG-PRM: Fine-grained hallucination detection and mitigation in language model mathematical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 4247–4278.
- Haolang Lu, Minghui Pan, Ripeng LI, Guoshun Nan, Jialin Zhuang, Zijie Zhao, Zhongxiang Sun, Kun Wang, and Yang Liu. 2026a. [Streaming hallucination detection in long chain-of-thought reasoning](#). In *Findings of the Association for Computational Linguistics: ACL 2026*.
- Haolang Lu et al. 2026b. Reallocating attention across layers to reduce multimodal hallucination. In *CVPR 2026*, pages 4157–4167.
- Liangchen Luo et al. 2024. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*.
- Uzay Macar, Paul C. Bogdan, Senthoooran Rajamanoharan, and Neel Nanda. 2026. Thought Branches: Interpreting LLM Reasoning Requires Resampling. In *Proceedings of the 14th International Conference on Learning Representations*.
- Amir Hameed Mir. 2025. The Geometry of Truth: Layer-wise Semantic Dynamics for Hallucination Detection in Large Language Models. *arXiv preprint*, arXiv:2510.04933.
- Shuaijie She, Junxiao Liu, Yifeng Liu, Jiajun Chen, Xin Huang, and Shujian Huang. 2025. R-PRM: Reasoning-Driven Process Reward Modeling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 13438–13451.
- Peiyang Song, Pengrui Han, and Noah Goodman. 2026. [Large language model reasoning failures](#). *Transactions on Machine Learning Research*. Survey Certification.
- Ponhvoan Srey, Xiaobao Wu, and Anh Tuan Luu. 2025. Unsupervised Hallucination Detection by Inspecting Reasoning Processes. In *EMNLP 2025*, pages 22117–22129.
- Zhongxiang Sun, Qipeng Wang, Haoyu Wang, Xiao Zhang, and Jun Xu. 2026. Mechanistic Detection and Mitigation of Hallucination in Large Reasoning Models. In *Proceedings of the 14th International Conference on Learning Representations*.
- Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging BIG-Bench Tasks and Whether Chain-of-Thought Can Solve Them. In *Findings of ACL*.
- Lexiang Tang, Xianwei Zhuang, Bang Yang, Zhiyuan Hu, Hongxiang Li, Lu Ma, Jinghan Ru, and Yuexian Zou. 2026. Not All Tokens and Heads Are Equally Important: Dual-Level Attention Intervention for Hallucination Mitigation. In *Proceedings of the 40th AAAI Conference on Artificial Intelligence*, pages 9439–9447.
- Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop Questions via Single-hop Question Composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. [Self-consistency improves chain of thought reasoning in language models](#). In *ICLR 2023*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*.
- An Yang, Anfeng Li, Baosong Yang, et al. 2025. Qwen3 technical report. *arXiv preprint*, arXiv:2505.09388.
- An Yang, Baosong Yang, et al. 2024. Qwen2.5 Technical Report. *arXiv preprint*, arXiv:2412.15115.
- Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit Panda, Jinyang Li, and He He. 2025a. Reasoning Models Know When They’re Right: Probing Hidden States for Self-Verification. In *COLM 2025*.
- Qiyuan Zhang et al. 2025b. A survey on test-time scaling in large language models: What, how, where, and how well? *arXiv preprint arXiv:2503.24235*.
- Zheng Zhao, Yeskendir Koishikenov, Xianjun Yang, Naila Murray, and Nicola Cancedda. 2026. Verifying chain-of-thought reasoning via its computational graph. In *ICLR 2026*.

A Probe Architecture and Training Details

We employ a simple, lightweight MLP probe architecture. The probe first maps input representations from dimension d to 256 using a linear projection. It then applies two hidden layers with dimensions $256 \rightarrow 64 \rightarrow 1$. Each hidden layer is followed by LayerNorm, ReLU activation, and dropout with a dropout rate of 0.3. The final layer uses a sigmoid activation. The input dimension d depends on the specific language model and problem setting.

Training is conducted for at most 100 epochs with batch size from 64 to 256 depending on the setup, using BCE loss. We use Adam with learning rate 3×10^{-4} and weight decay 1×10^{-5} . A WeightedRandomSampler is used to balance positive and negative samples. The learning rate is scheduled by ReduceLROnPlateau in max mode, with patience 8 and reduction factor 0.5. All runs use a fixed random seed of 42.

During training, we observe only minimal changes in performance when varying hyperparameters. We therefore opt for this simple configuration, as it encourages the probe to capture genuine error-related signals while reducing reliance on fine-grained artifacts that more expressive architectures may exploit.

B Data Details

B.1 Annotation Quality of the CHAINED Dataset

CHAINED labels are generated through a hybrid pipeline combining automated filtering with expert validation. Lu et al. (2026a) establish a three-stage procedure: (i) an answer-aware semantic check performed by a frontier model—Claude 4.5; (ii) a rule-based logical consistency filter that discards invalid state transitions, such as trace recovery without corrective intervention; and (iii) manual expert review of a stratified sample, achieving over 96% agreement with the automated annotations. Consequently, we treat e_t and c_t as highly reliable, albeit imperfect, supervisory signals.

B.2 Unrecoverable vs Recoverable Reasoning Traces Generation

This section details the automated data generation pipeline used to construct the reasoning traces data used in Section 6. Inspired by Luo et al. (2024), we employ a Monte Carlo (MC) estimation and an Upper Confidence Bound (UCB) guided binary search

to pinpoint the exact onset of reasoning failures ($\hat{t}_{\text{unre}}^*$) from the base dataset prompts.

Pipeline Setup. We utilize the exact same set of prompt questions from the CHAINED dataset (Lu et al., 2026a). Beyond evaluating the three original baseline models, we furthermore expand our generation pipeline to include three recent small-size to mid-size models: Qwen3-4B, Qwen3-8B, and Qwen3-14B. All generation experiments are conducted on H100 GPUs hosted via vLLM.

Prompts We present prompts used to generate reasoning trajectories for all of our experiments in Table 7. For DSR8B, Lla8B, and Qwe7B, we follow the generation setup introduced in the CHAINED dataset. For the Qwen3 model family, we utilize a common, simplified prompt template. This is because Qwen3 models are post-trained to naturally output structured, step-by-step reasoning traces without heavy prompt engineering.

1. Root Monte Carlo Estimation. To assess prompt viability, we generate initial 12 completions from an empty prefix. The empirical root recovery probability $\hat{\rho}_{\text{root}}$ is calculated as:

$$\hat{\rho}_{\text{root}} = \frac{1}{K_{\text{valid}}} \sum_{k=1}^{K_{\text{valid}}} \mathbb{I}(y^{(k)} = y^*)$$

where K_{valid} is the number of scorable (non-truncated) rollouts, and y^* is the ground-truth answer. We retain only prompts exhibiting mixed outcomes ($0 < \hat{\rho}_{\text{root}} < 1$), ensuring the model possesses partial capability to solve the problems.

2. Candidate Selection via UCB Scoring. For retained prompts, we systematically search for the earliest unrecoverable step. We select candidate trajectories from the pool of incorrect rollouts using a UCB-style scoring function that prioritizes low MC estimates, unexplored states, and longer continuations:

$$\text{Score}(c) = Q(c) + U(c) + B(c)$$

where the components are defined as:

- **Quality Score (Q):** $Q(c) = \alpha^{(1-\hat{\rho}_{\text{root}})} \cdot \beta^{(L_c/L_{\text{max}})}$
- **Exploration Score (U):** $U(c) = c_{\text{puct}} \frac{\sqrt{N_{\text{total}}}}{1+N_c}$
- **Length Bonus (B):** $B(c) = w_{\text{step}} \log(1 + L_c)$

Here, L_c is the number of continuation steps, N_c is the state visit count, and N_{total} is the total visits across all states. We set hyperparameters $\alpha = 0.5$, $\beta = 0.9$, $L_{\text{max}} = 200$, $c_{\text{puct}} = 0.125$, and $w_{\text{step}} = 0.3$.

Prompt Template for DSR8B, Lla8B, and Qwe7B

System Prompt:

You are a hyper-rigorous, expert logician and microscopic reasoner. Your task is to produce an **EXHAUSTIVE and HYPER-DETAILED Chain-of-Thought (CoT)**. **DECOMPOSE the problem into the SMALLEST POSSIBLE ATOMIC STEPS**. Your reasoning must consist of **MAXIMUM unique, non-redundant steps**. Each step must be a single, short logical jump. Do not combine micro-inferences. Focus on **UNPACKING** every implicit connection and detail, making the reasoning process as long and granular as logically possible. Crucially, your thinking process must not contain repeated, cyclical, or excessively similar steps. Always conclude with 'Final Answer: ' followed by your final, clear answer.

User Template:

TASK: Solve the following complex riddle or question.

INPUT QUESTION:

{QUESTION}

INSTRUCTIONS:

1. Produce a numbered list of reasoning steps (1., 2., 3., ...).
 2. Start by detailing your step-by-step reasoning (Chain-of-Thought). **Your goal is to generate as many unique, non-repetitive steps as possible to fully decompose the problem**
 3. Each step should be ONE short sentence performing ONE single micro-inference (atomic). **For example, a typical human thought might take 3-5 of your atomic steps to fully explain.**
 4. **Deeply decompose the problem:** Assume the reader knows nothing, and explicitly state every piece of information used and every trivial logical deduction made. Your goal is to achieve the greatest possible number of unique, necessary, atomic steps.
 5. Do not skip intermediate hops; unfold implicit links explicitly.
 6. Ensure every step is unique and directly advances the solution. **If you find yourself stuck in a loop or repeating steps, STOP the reasoning immediately and proceed directly to the final answer.**
 7. Conclude your entire response with the final answer, and it must be prefixed by the exact string: 'Final Answer: '.
 8. **STOP GENERATING** any further text or reasoning immediately after the 'Final Answer:' line.
-

Prompt Template for Qwen3 Models

System Prompt:

(None)

User Template:

Please answer the following question.

You should provide your final answer in the format \boxed{{YOUR_ANSWER}}.

Separate your following steps using \n\n.

Question:

{QUESTION}

Table 7: System prompts and user templates used for generating reasoning trajectories across models.

3. Trace Slicing via Binary Search. Once a candidate trajectory is selected, we locate the exact failure point using a binary search over the reasoning steps, formalizing the OmegaPRM (Luo et al., 2024) as in Algorithm 1. The budget K for this phase is set to 16, ϵ is set to 0.0:

If Algorithm 1 only identifies errors at the final step ($t^* = T$), a retry phase (up to 6 extra iterations) is triggered on longer continuations to actively hunt for mid-step errors.

4. Stricter Re-search for Early Failure Detection.

The base search utilizes a zero recovery threshold ($\epsilon = 0$), meaning a single correct rollout out of K marks a prefix as valid. Under this lenient setting, we observe that for many traces ($\approx 15 - 25\%$) of generated samples, the identified unrecoverable point $\hat{t}_{\text{unre}}^*$ is located at the very end of the trajec-

tory (i.e., at step $T - 1$ or T). To rigorously isolate earlier structural breakdowns and achieve a more diverse distribution of error locations, we apply a stricter re-search phase specifically for these late-stage failure trajectories (where $t^* \geq T - 1$).

During this phase, we tighten the viability threshold to $\epsilon = 0.0625$ (requiring $> 1/16$ correct rollouts for a prefix to be considered recoverable). If this stricter threshold is violated earlier in the trace, the designated error point $\hat{t}_{\text{unre}}^*$ is shifted to the newly discovered earlier step.

5. Dataset Balancing and Generation Setup.

To construct a balanced trace-level dataset for classification, we aggregate the identified unrecoverable traces with fully correct trajectories. Then, correct traces are gathered by sampling additional rollouts from the retained prompts to serve as nega-

Algorithm 1 Binary Search for $\hat{t}_{\text{unre}}^*$

Require: Trajectory $(s_1, \dots, s_T)$, Policy π_θ , Budget K , Threshold ϵ

- 1: **Initialize:** low $\leftarrow 1$, high $\leftarrow T$, $t^* \leftarrow T$
- 2: **while** low $\leq$ high **do**
- 3: mid $\leftarrow \lfloor (\text{low} + \text{high})/2 \rfloor$
- 4: $p_{\text{mid}} \leftarrow (s_1, \dots, s_{\text{mid}})$ $\triangleright$ Extract prefix
- 5: Generate K rollouts from p_{mid} using π_θ
- 6: $\hat{\rho}_{\text{mid}} \leftarrow \frac{1}{K} \sum_{i=1}^K \mathbb{I}(y^{(i)} = y^*)$ $\triangleright$ Compute empirical recovery
- 7: **if** $\hat{\rho}_{\text{mid}} > \epsilon$ **then**
- 8: low $\leftarrow \text{mid} + 1$ $\triangleright$ Prefix is recoverable, search right
- 9: **else**
- 10: $t^* \leftarrow \text{mid}$
- 11: high $\leftarrow \text{mid} - 1$ $\triangleright$ Prefix is unrecoverable, search left
- 12: **end if**
- 13: **end while**
- 14: **return** t^*

Model	Total	# wrong traces	$\hat{t}_{\text{unre}}^*$ rel position
Llama-3.1-8B-Instruct	5,693	3,700	0.701 ± 0.231
Qwen2.5-7B-Instruct	4,611	2,647	0.694 ± 0.246
DeepSeek-R1-Distill-Llama-8B	4,522	2,498	0.654 ± 0.258
Qwen3-14B	3,666	1,445	0.670 ± 0.263
Qwen3-8B	4,304	1,932	0.632 ± 0.274
Qwen3-4B	5,781	1,858	0.595 ± 0.284

Table 8: Summary statistics of the generated trace-level dataset across models, detailing the total number of sampled traces, the subset of incorrect trajectories, and the average relative position of the unrecoverable boundary ($\hat{t}_{\text{unre}}^*$).

tive data samples. All generations in this pipeline utilize a temperature of 0.6 and $\text{top}_p = 0.95$, with a maximum output length of 20,000 tokens. Final answers are automatically extracted via structural heuristics (e.g., `\boxed{...}` or `<think>` tags) and evaluated using exact string matching and numeric equivalence checks. The final summary statistics of the generated dataset across all evaluated models are shown in Table 8.

C Additional Experimental Results

C.1 Extended Results on Local Feature Ensembling

Table 9 presents the extended AUC results using larger window sizes for Table 3 in Section 5, specifically $w \in \{4, 5\}$.

As the window size expands to 4 and 5 steps, the average performance drop (Avg. Δ) for both pooling and concatenation configurations continues

Probing Features	DSR8B	Lla8B	Qwe7B	Avg. Δ
$\mathbf{h}_t$ only	.922	.851	.814	–
pool ($\mathbf{H}_{t,1}$)	.886	.813	.776	-.037
concat ($\mathbf{H}_{t,1}$)	.923	.836	.797	-.011
pool ($\mathbf{H}_{t,1}^{-t}$)	.876	.793	.758	-.053
concat ($\mathbf{H}_{t,1}^{-t}$)	.903	.802	.766	-.039
pool ($\mathbf{H}_{t,2}$)	.878	.797	.754	-.053
concat ($\mathbf{H}_{t,2}$)	.924	.830	.788	-.015
pool ($\mathbf{H}_{t,2}^{-t}$)	.872	.783	.741	-.064
concat ($\mathbf{H}_{t,2}^{-t}$)	.910	.799	.759	-.040
pool ($\mathbf{H}_{t,3}$)	.870	.785	.732	-.067
concat ($\mathbf{H}_{t,3}$)	.920	.826	.771	-.023
pool ($\mathbf{H}_{t,3}^{-t}$)	.865	.774	.721	-.076
concat ($\mathbf{H}_{t,3}^{-t}$)	.906	.797	.746	-.047
pool ($\mathbf{H}_{t,4}$)	.858	.777	.713	-.080
concat ($\mathbf{H}_{t,4}$)	.912	.822	.757	-.032
pool ($\mathbf{H}_{t,4}^{-t}$)	.854	.767	.702	-.088
concat ($\mathbf{H}_{t,4}^{-t}$)	.897	.795	.732	-.054
pool ($\mathbf{H}_{t,5}$)	.852	.772	.705	-.086
concat ($\mathbf{H}_{t,5}$)	.909	.819	.736	-.041
pool ($\mathbf{H}_{t,5}^{-t}$)	.849	.763	.695	-.093
concat ($\mathbf{H}_{t,5}^{-t}$)	.893	.792	.715	-.062

Table 9: AUC results for predicting the current step’s label w.r.t. Full-window Ensemble (including $\mathbf{h}_t$) vs. Context-only Ensemble (excluding $\mathbf{h}_t$). Here, $\mathbf{H}_{t,w} = \{\mathbf{h}_{t+k} \mid -w \leq k \leq w\}$ denote the full window of $2w + 1$ step representations centered at $\mathbf{h}_t$, and $\mathbf{H}_{t,w}^{-t} = \mathbf{H}_{t,w} \setminus \{\mathbf{h}_t\}$ denote the corresponding context-only window. pool() averages the step representations; while concat() concatenates them. “Avg. Δ ” reports the average AUC change relative to the baseline of using $\mathbf{h}_t$ alone as input, averaged across all three models.

to widen. This trend further reinforces our finding in the main text that local error signals share the same footprint of error to local window of steps.

C.2 Length-Controlled Trace-Level Signal Experiments

Motivation and Setup In Section 6, we demonstrated that final hidden states can distinguish between recoverable (R) and unrecoverable (U) errors. However, generation length can often act as a confounding artifact in reasoning evaluation. As detailed in Table 10, there is a notable length discrepancy between the two error classes in the original dataset, especially in three base models (DSR8B, Qwe7B, and Lla8B). To ensure that the probe learns genuine representational signal rather than trace length’s confounding factor, we conduct an ablation study using a length-matched dataset.

For each model, we calculate the total character length of every trace (summing across all step texts) and divide the length distribution into 20 log-spaced bins. Within each bin, we randomly subsample an equal number of R and U traces (i.e., $\min(n_R, n_U)$). This matching procedure is performed independently across the standard train, validation, and test splits (70% / 15% / 15%), ensuring an identical length distribution between the two classes (length ratio ≈ 1.00). The resulting balanced dataset sizes per class are detailed in Table 11.

The binary classification probe is then trained following the setup described in Appendix A.

Model	Len (R)	Len (U)	Overall Len	Ratio (U/R)
DSR8B	14,690	23,792	19,241	1.62
QWEN7B	1,911	2,839	2,375	1.49
LLA8B	15,165	10,373	12,769	0.68
QWEN3-4B	8,140	11,160	9,650	1.37
QWEN3-8B	8,086	9,801	8,944	1.21
QWEN3-14B	7,085	7,913	7,499	1.12

Table 10: Average trace length (Len, in characters) for recoverable (R) and unrecoverable (U) trajectories prior to length matching, alongside the U/R length ratio.

Model	Train	Val	Test	Total Pairs	Avg Len
DSR8B	648	120	125	893	16,884
QWEN7B	459	92	88	639	1,970
LLA8B	223	42	38	303	10,282
QWEN3-4B	1018	221	211	1450	9,320
QWEN3-8B	1084	213	207	1504	9,226
QWEN3-14B	866	175	171	1212	7,477

Table 11: Number of samples per class ($R = U$) across splits after trace-length matching, with average trace length (chars). The total number of traces in each split is exactly $2 \times$ the reported number.

Model	AUC	F1
DSR8B	.824	.770
QWE7B	.820	.768
LLA8B	.844	.763
QWEN3-4B	.898	.798
QWEN3-8B	.880	.821
QWEN3-14B	.847	.776

Table 12: Trace-level classification performance ($R \leftrightarrow U$) evaluated on the strictly length-matched test set.

Results As shown in Table 12, even when completely isolated from length-based confounds, the final hidden states retain strong separability. The probe maintains high classification performance across all evaluated models, e.g., achieving an AUC of 0.898 on Qwen3-4B and 0.880 on Qwen3-8B.

R	ϵ	AUC	F1
16	0	.823	.740
32	0	.855	.750
48	0	.864	.786
16	.0625	.788	.715
16	.1250	.742	.701
16	.1875	.751	.708

Table 13: Sensitivity of Recoverable $\leftrightarrow$ Unrecoverable classification on Qwe7B to the resampling budget R and unrecoverability threshold ϵ . We vary R at $\epsilon = 0$ and vary ϵ using the base budget $R = 16$.

Notably, these results are highly competitive and closely match the original unmatched performance, with Qwe7B (AUC = .820) even surpassing the original baseline (.768). This firmly suggests that the distinction between recoverable and unrecoverable errors is encoded as a qualitative representational change in the latent space.

C.3 Robustness of Recoverability Labels

To assess the robustness of our empirical recoverability labels in Section 6, we perform additional experiments by varying both the resampling budget R and the unrecoverability threshold ϵ . Due to computational constraints, we conduct this robustness analysis on Qwe7B.

Table 13 shows that the representational distinction remains robust across these settings. At $\epsilon = 0$, increasing the resampling budget from $R = 16$ to 32 and 48 steadily improves separability, with AUC increasing from .823 to .855 and .864, respectively. Increasing ϵ under the base budget $R = 16$ makes the unrecoverability criterion less strict and consequently reduces AUC, but performance remains above .74 across all evaluated thresholds.

We further evaluate the stability of prefixes labeled unrecoverable under the original $R = 16$, $\epsilon = 0$ configuration by resampling each prefix with $R = 48$. Among these prefixes, **87.2%** still yield zero successful continuations, while **9.4%** yield only 1–6 successful continuations out of 48 and **3.3%** yield more than 6. Thus, the strict zero-success label is preserved for the large majority of examples under a three-fold larger sampling budget, while most remaining prefixes still exhibit low empirical recoverability. For a prefix with zero successes in 48 independent rollouts, the one-sided 95% Clopper–Pearson upper confidence bound on its true recovery probability is 0.061.

C.4 Retrospective Tracing via Representational Differences.

Model	k	Ranking		Recall			AUC
		MRR	bMRR	@1	@3	@5	
DSR8B	-3	.177	.627	1.7	.200	.333	.783
	-2	.213	.605	5.0	.250	.367	.801
	-1	.246	.620	8.3	.300	.433	.785
	0	.428	.761	28.3	.483	.600	.883
	+1	.302	.717	16.7	.286	.452	.853
	+2	.206	.627	5.0	.250	.300	.814
	+3	.270	.634	13.2	.316	.395	.830
	Rand.	.208	.302	6.5	.195	.325	—
QWE7B	-3	.135	.694	1.7	.075	.167	.785
	-2	.146	.775	1.9	.067	.169	.817
	-1	.193	.771	3.6	.167	.303	.852
	0	.332	.873	16.4	.342	.525	.925
	+1	.244	.839	7.7	.252	.390	.911
	+2	.251	.799	8.0	.255	.433	.894
	+3	.285	.788	12.3	.262	.440	.895
	Rand.	.198	.304	5.8	.174	.290	—
LLa8B	-3	.134	.638	2.5	.090	.165	.747
	-2	.184	.697	6.5	.158	.208	.784
	-1	.187	.717	5.0	.189	.289	.809
	0	.330	.787	15.8	.385	.512	.896
	+1	.243	.755	9.6	.231	.373	.859
	+2	.226	.715	9.4	.216	.324	.856
	+3	.207	.669	5.8	.204	.335	.848
	Rand.	.175	.283	5.0	.149	.248	—

Table 14: Source error tracing performance across temporal offsets k utilizing representational difference features ($\mathbf{h}_T - \mathbf{h}_{t_{\text{pers}}+k}^*$). Best per model in **bold**.

To further isolate the nature of the persistent error signal, we conduct an ablation study where the joint feature representation is formulated by the vector difference rather than concatenation. Specifically, the input to the MLP probe is defined as $\mathbf{h}_T - \mathbf{h}_{t_{\text{pers}}+k}^* \in \mathbb{R}^d$, representing the directional difference from an intermediate t -th step to the last step’s representation.

As shown in Table 14, utilizing the representational difference vector still yields strong localization performance that peaks precisely at the true failure origin ($k = 0$), nearly matches the concatenation version and vastly outperforming the other baselines. This robust performance demonstrates that the directional change carries a highly informative signal of the persistent error.

D Qualitative Examples of Retrospective Tracing

Figure 2 illustrates that the probe scores $f(h_T, h_t)$ exhibit diverse patterns across different models and trajectories. Despite this global diversity, a consistent observation is that the scores for steps strictly

prior to t_{pers}^* are typically very low, often near zero. For example, in the Qwe7B trace (bottom left), the predicted probability remains flat at approximately 0.0 for all steps from $t = 0$ to $t = 10$.

Furthermore, the probe frequently outputs multiple prominent spikes in the immediate nearby steps of t_{pers}^* . For instance, in the DSR8B trace (top right), while the highest peak aligns with $t_{\text{pers}}^* = 8$ (≈ 0.91), the nearby steps also show strong signals, such as $t = 10$ (≈ 0.80), $t = 12$ (≈ 0.65). Beyond these nearby spikes, the patterns remain diverse toward the end of the traces: some trajectories maintain fluctuating high scores (e.g., the Lla8B trace on the middle left, bouncing between 0.4 and 0.9 after $t = 12$), while others exhibit sharp drops before spiking again at the final step.

E Token-Efficient Prefix Resampling

To evaluate the efficiency of our probe-guided decoding, we compare our approach against the following baselines. For all sampling-based methods, we set the generation budget to $N = 5$ trajectories:

- **Greedy (Base):** Standard greedy decoding from the base model, serving as a minimal baseline.
- **Self-Consistency (SC):** Majority voting over independently sampled candidate solutions (Wang et al., 2023).
- **Best-of- N (BoN):** The policy model samples N candidate responses, which are then evaluated by our trace-level probe. We select the trajectory with the lowest predicted error score (Brown et al., 2024).
- **Weighted SC:** A modified SC approach where each candidate’s vote in the majority consensus is weighted by its corresponding trace-level probe score.
- **Prefix Resampling:** A token-efficient strategy utilizing both our probes. First, the trace-level probe evaluates the initial generation. If an error is detected, the retrospective tracing probe (Section 7) identifies the persistent error step by retrieving the top-5 steps with the highest error probabilities and selecting the one with the smallest step index (i.e., the earliest detected failure). The trajectory is truncated just before this step, and generation resumes from the uncontaminated prefix. This process loops until the trace-level error score falls below a predefined safety threshold or the maximum budget ($N = 5$) is reached.

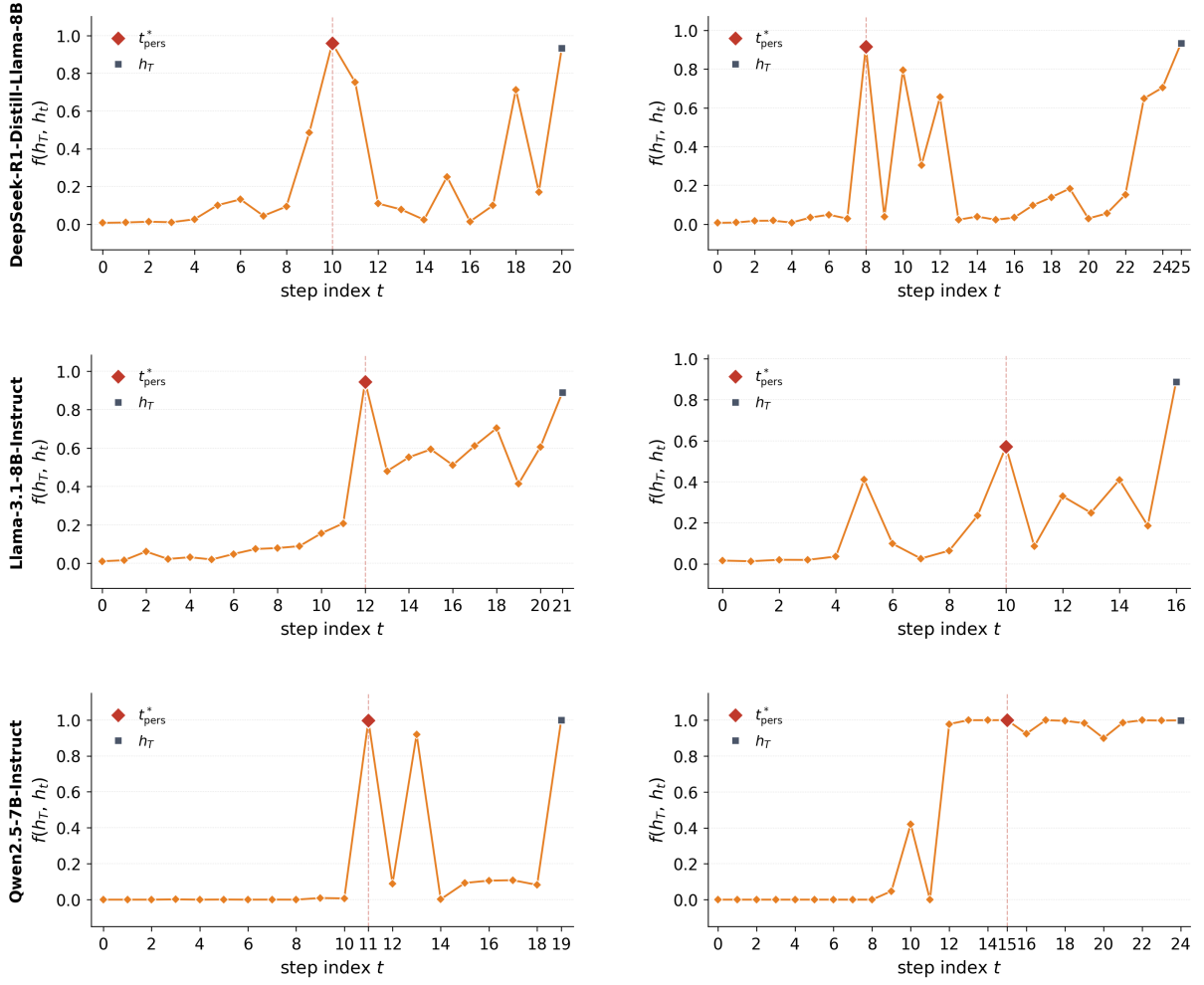


Figure 2: Qualitative examples of retrospective tracing across six different reasoning trajectories from three models. The x-axis represents the reasoning step index t , and the y-axis shows the normalized probe prediction score $f(h_T, h_t)$ estimating the likelihood of step s_t being the persistent error step. The red dashed line and diamond marker denote the true source of error propagation (t_{pers}^*), while the blue square indicates the final reasoning step (h_T).