THE ERROR YOU SEE IS NOT THE ERROR YOU MADE: PROGRESSION-AWARE REASONING ORIGIN FOR REASONING ERROR LOCALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Verifying multi-step LLM reasoning requires more than determining whether a trace is correct: a useful verifier should identify where the reasoning first goes wrong. However, existing holistic methods provide little positional evidence, while forward sequential verification often treats the first rejected step as the error source. Under error propagation, this assumption can fail, since an earlier mistake may remain locally plausible and become observable only through its downstream consequences. We therefore *rethink reasoning verification as a progression-aware error-source localization problem*: rather than asking only where a reasoning trace first appears inconsistent, we ask which earlier step best explains how that inconsistency emerges along the trajectory. Based on this view, we propose **Progression-aware Reasoning Origin (PRO)**, a training-free framework for first-error localization. PRO jointly models incoming support from the preceding context and outgoing compatibility with subsequent reasoning, selectively refines regions where these signals disagree, and finally performs detector-conditioned source attribution with intervention-based evidence to distinguish the true error origin from its propagated manifestations. We further formalize the gap between forward rejection and structural exposure, showing why incoming-side evidence alone is insufficient for reliable localization under error propagation. Experiments across open-form, medical, and structured reasoning tasks demonstrate consistent improvements over strong verification baselines, supporting progression-aware source attribution as a more faithful formulation of reasoning verification.

1 INTRODUCTION

Large language models (LLMs) are increasingly capable of solving complex multi-step reasoning tasks, driven in part by advances in chain-of-thought prompting and zero-shot reasoning (Wei et al., 2022; Kojima et al., 2022), decoding-time aggregation such as self-consistency (Wang et al., 2023), and more structured inference-time search paradigms (Yao et al., 2023; Besta et al., 2024). Yet verifying their reasoning remains difficult. Final-answer checking is insufficient because a trace can contain invalid intermediate steps even when it reaches a correct answer. A useful verifier must therefore determine not only whether a trace is flawed, but also which step first introduces the error.

Existing verification methods mainly follow two paradigms. Related answer-level self-verification and critique frameworks further show the practical value of test-time checking, but they still focus primarily on improving or validating complete responses rather than localizing the earliest corrupted step (Weng et al., 2023; Gou et al., 2024). Holistic verifiers assess the complete chain and return a single judgment, providing little positional evidence for error attribution. Sequential verifiers instead decompose a trace into smaller units and validate them step by step, improving interpretability and control. However, standard sequential verification is driven primarily by forward local checking: it asks whether the current block is supported by the problem and the accepted prefix. Under error propagation, the first block rejected by this procedure can be a downstream exposure point rather than the step that introduced the error. A locally plausible block may become suspicious only when its consequences appear in later reasoning, creating a gap between error exposure and error attribution.

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.

When errors propagate through a reasoning trace, how can a verifier localize the first erroneous step rather than a later step at which the error becomes observable?

Addressing this question requires moving beyond verification as a sequence of independent accept-or-reject decisions. Reasoning errors are inherently *progressive*: once introduced, they can distort subsequent context while leaving later steps locally coherent with a corrupted trajectory. Consequently, the first observable failure may occur after the true error source, creating a gap between error origin and error exposure. Reliable localization therefore requires reasoning not only about local validity, but also about how verification evidence evolves along the trajectory. Downstream consequences can provide retrospective evidence that exposes earlier mistakes, while later rejected steps may merely reflect propagated errors. This motivates explicitly modeling the progression of verification evidence to distinguish where an error originates from where it first becomes observable.

Motivated by the progression-aware view, we propose **Progression-aware Reasoning Origin (PRO)**, a training-free framework for progression-aware error-source localization in decomposed reasoning traces. PRO treats each reasoning step as both a conclusion supported by its preceding context and a premise that should remain compatible with its subsequent continuation. Accordingly, it evaluates complementary *incoming* and *outgoing* evidence: the former measures whether a step is justified by the problem and the verified prefix, while the latter examines whether the step can consistently support the reasoning that follows. Their agreement provides stronger evidence for localization, whereas disagreement indicates a structurally ambiguous region in which the exposed failure may not coincide with the true source.

Rather than making an immediate decision at such ambiguous positions, PRO progressively narrows the candidate source. It first performs coarse-grained trajectory-level verification to identify suspicious regions, then selectively refines the reasoning granularity only where the evidence remains inconclusive. Finally, conditioned on the observed verification pattern, PRO compares the remaining source candidates to distinguish the step that introduces the error from downstream steps that merely inherit or expose it. This coarse-to-fine procedure enables PRO to exploit both local consistency and downstream consequences while avoiding the forward-only assumption that the earliest rejected step is necessarily the error origin. Our main contributions can be summarized as:

- We formulate reasoning verification as a first-error localization problem, and argue that accurate localization requires more than purely forward local checking.
- We propose PRO, a progression-aware verification framework that combines backward consistency checking, mismatch-triggered fine-grained refinement, and top-k candidate localization.
- We formalize the discrepancy between forward rejection and structural exposure under error propagation, providing a theoretical basis for incorporating outgoing evidence in earliest-error localization.

2 Related Work

Reasoning Verification. Existing verification methods are mainly holistic or decomposition-based. Early verifier work learns critics to score complete sampled solutions and rerank candidates (Cobbe et al., 2021; Uesato et al., 2022). Recent methods move toward finer process assessment by inspecting steps, blocks, or explicit structural relations. GoV organizes verification units into graph-structured blocks with adaptive granularity (Fang et al., 2026), while ProcessBench and DeltaBench provide earliest-error benchmarks and show that process-error detection remains difficult even for strong PRMs and critic-style verifiers (Zheng et al., 2025; He et al., 2025).

Process Supervision. Process supervision provides step-level feedback rather than relying only on final outcomes. Uesato et al. (2022) compare process- and outcome-based supervision for mathematical reasoning, and Lightman et al. (2023) establish the modern PRM paradigm with PRM800K. Subsequent work reduces annotation cost through automated process labeling, as in Math-Shepherd (Wang et al., 2024) and OmegaPRM (Luo et al., 2024). Together, these methods make dense intermediate supervision a standard tool for verifier training, reranking, and reinforcement learning.

Self-Correction and Test-Time Verification. Another related line improves reasoning by critique, revision, and verification at inference time. Representative examples here include Self-Refine

(Madaan et al., 2023), Chain-of-Verification (Dhuliawala et al., 2024), ProCo (Wu et al., 2024), and Derailer-Rerailer (Wan et al., 2025). At the same time, recent analysis also shows that intrinsic self-correction remains limited without strong external feedback or explicit verification signals (Huang et al., 2023). These methods mainly focus on improving or stabilizing generated reasoning, rather than locating the earliest erroneous step under explicit process annotations.

3 Detector-Stage Gap in Sequential Verification

3.1 Incoming Admissibility and Structural Exposure

The limitation of forward-only verification arises from the evidence available at each block. A standard sequential verifier evaluates b_i only from its incoming context, namely the problem q and the accepted prefix $\mathcal{P}_{i-1}$. Consequently, a block may remain locally admissible even when it introduces a defect whose inconsistency becomes visible only through the subsequent transition. In such cases, the earliest forward rejection occurs after the underlying error has already entered the trajectory. To characterize this gap, we distinguish incoming admissibility from outgoing compatibility.

Specifically, we define $I_i = \mathbf{1}[(q, \mathcal{P}_{i-1}) \models b_i], O_i = \mathbf{1}[b_i \vdash b_{i+1}], i < n$, where $\models$ denotes semantic-logical support from the problem and preceding reasoning, and $b_i \vdash b_{i+1}$ denotes whether b_i provides a valid basis for the next reasoning block. Thus, I_i characterizes whether b_i is admissible from its incoming context, whereas O_i characterizes whether its local continuation remains structurally compatible with the subsequent reasoning.

We then define the mismatch set $\mathcal{M}(q, \mathcal{B}) = \{\, i < n \mid I_i = 1,\ O_i = 0 \,\}$, which contains blocks that appear admissible under forward checking but fail to support their immediate continuation. These blocks are precisely the cases that cannot be exposed from prefix evidence alone.

Under an ideal forward-only verifier, the earliest rejection index is $\tau_{\text{seq}}(q, \mathcal{B}) = \min\big(\{\, i \mid I_i = 0 \,\} \cup \{\infty\}\big)$. By contrast, once outgoing compatibility is also considered, the earliest detector-stage exposure becomes $\tau_{\text{exp}}(q, \mathcal{B}) = \min\big(\{\, i \mid I_i = 0 \,\} \cup \mathcal{M}(q, \mathcal{B}) \cup \{\infty\}\big)$.

Importantly, τ_{exp} is not yet the final first-error localization $f(q, \mathcal{B})$. It only identifies the earliest block at which either an incoming inconsistency or an outgoing incompatibility becomes detectable. The exposed block may itself be erroneous, or it may reflect the downstream consequence of an earlier error. A subsequent source-attribution stage is therefore required to determine which preceding block first introduced the inconsistency.

3.2 Formal Properties of the Detector-Stage Gap

The definitions above nevertheless reveal a fundamental limitation of forward-only detection. If outgoing compatibility provides no additional information, then τ_{exp}reduces to the ordinary forward-rejection index τ_{seq}. However, whenever a block remains admissible from its prefix but fails to support the subsequent transition, the outgoing-side test can expose a suspicious location before any forward rejection occurs. This suggests two questions: whether such earlier exposure is formally guaranteed to be no later than forward rejection, and whether it can be recovered from incoming-side evidence alone. The following results answer these questions.

Theorem 1 (Strict Gap Between Forward Rejection and Structural Exposure)**.** *To make the comparison with ordinary forward rejection explicit, the ordering relation can be formulated as:*

$$\tau_\star(q, \mathcal{B}) \leq \tau_{\text{seq}}(q, \mathcal{B}). \tag{1}$$

Moreover, there exist q and $\mathcal{B}$ such that the inequality is strict.

Theorem 2 (Necessity of Outgoing-Side Information)**.** *To show why incoming-side evidence alone cannot determine structural exposure, we consider two traces that are indistinguishable from prefix-only signals. Their separation can be expressed as:*

$$\forall k,\ I_k(\mathcal{B}) = I_k(\mathcal{B}'), \qquad \tau_\star(q, \mathcal{B}) \neq \tau_\star(q, \mathcal{B}'). \tag{2}$$

Consequently, no detector rule that depends only on incoming-side signals $\{I_k\}$ can recover $\tau_\star$ correctly for all decomposed reasoning traces.

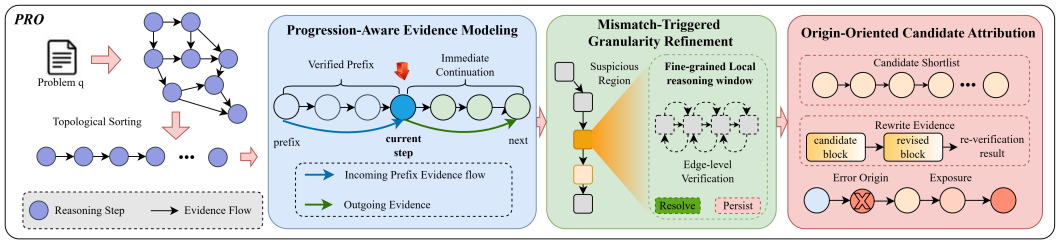

Figure 1: The overview of the proposed PRO.

These results do not by themselves recover the final source target $f(q, \mathcal{B})$. Instead, they motivate a forward-anchored detector augmented with an outgoing-side structural test. PRO subsequently refines these pre-refinement exposure candidates before using the surviving detector output for source localization.

4 Progression-aware Reasoning Origin

4.1 Overview

In this paper, we revisit reasoning verification from a *progression-aware* perspective. Beyond checking whether each block is supported by its preceding context, we also examine its compatibility with subsequent reasoning, allowing downstream inconsistency to provide retrospective evidence for distinguishing error origins from propagated failures. Motivated by this, we propose a training-free framework PRO for first-error localization. As illustrated in Figure 1, PRO consists of three stages. First, **progression-aware evidence modeling** jointly examines the incoming support of each block and its compatibility with subsequent reasoning to identify potential progression inconsistencies. Second, **mismatch-triggered granularity refinement** selectively re-examines ambiguous regions at a finer reasoning granularity, reducing uncertainty introduced by coarse block decomposition and local transition noise. Third, **origin-oriented candidate attribution** uses the refined exposure position as a detector anchor, constructs a detector-conditioned shortlist, and further compares these candidates with rewrite-supported evidence, thereby tracing the observed failure back to its most plausible error origin. Through this progression from inconsistency detection to ambiguity resolution and source attribution, PRO explicitly separates *where an error becomes observable* from *where it is introduced*.

4.2 Progression-aware Evidence Modeling

As discussed above, conventional sequential verification relies primarily on forward local checking, and therefore tends to identify the earliest block at which an error becomes observable rather than the block that actually introduces it. Under error propagation, these two positions may differ: an earlier erroneous block can remain locally consistent with the accepted prefix, while its consequence is only exposed in subsequent reasoning.

To bridge this gap, PRO adopts a *progression-aware* view of verification. Rather than judging each block solely from its preceding context, PRO considers its role within the local reasoning progression. For a block b_i, evidence can be examined in two directions. From the preceding context to b_i, we ask whether the current block is a valid conclusion supported by the problem and the verified prefix; we refer to this as *incoming evidence*. From b_i to the subsequent reasoning, we ask whether the current block can serve as a valid premise for the continuation; we refer to this as *outgoing evidence*. Together, the two signals characterize whether a block is locally justified while remaining consistent with the reasoning progression.

Incoming evidence. Let $\mathcal{P}_{i-1}$ denote the verified prefix before b_i. We define $V_{\text{in}}(b_i) = \text{Verify}_{\text{in}}\big(b_i; q, \mathcal{P}_{i-1}\big)$, where $V_{\text{in}}(b_i) \in \{0, 1\}$ indicates whether b_i is justified by the problem and its accepted prefix. This corresponds to the ordinary forward-anchored verification signal.

Outgoing evidence. For $i < n$, PRO further evaluates whether the same block can serve as a valid premise for the immediate continuation:

$$V_{\text{out}}(b_i) = \text{Verify}_{\text{out}}\big(b_i; q, b_{i+1}\big). \tag{3}$$

Rather than reversing the full reasoning chain, this test examines whether the local progression from b_i to b_{i+1} remains coherent. For the terminal block, we set $V_{\text{out}}(b_n) = 1$.

The incoming and outgoing signals jointly define a progression-aware verification state for each block. Specifically, we introduce:

$$s_i = \begin{cases} \text{R}, & V_{\text{in}}(b_i) = 0, \\ \text{A}, & V_{\text{in}}(b_i) = 1 \ \wedge \ V_{\text{out}}(b_i) = 1, \\ \text{M}, & V_{\text{in}}(b_i) = 1 \ \wedge \ V_{\text{out}}(b_i) = 0, \end{cases} \tag{4}$$

where R, A, and M denote *rejection*, *agreement*, and *mismatch*, respectively. The corresponding index sets are $\mathcal{R} = \{i : s_i = \text{R}\}$, $\mathcal{A} = \{i : s_i = \text{A}\}$, and $\mathcal{M} = \{i : s_i = \text{M}\}$, respectively.

This partition reflects the asymmetric roles of the two verification signals. A block enters $\mathcal{R}$ as soon as its incoming evidence fails, indicating that the inconsistency is already exposed under forward verification. When the incoming evidence is positive, the outgoing signal further distinguishes between progression agreement and progression mismatch. Thus, $\mathcal{A}$ contains blocks that are supported by both their preceding context and subsequent progression, whereas $\mathcal{M} = \{i : V_{\text{in}}(b_i) = 1, V_{\text{out}}(b_i) = 0\}$ captures the cases in which forward verification alone remains insufficient. We refer to these positions as *progression mismatches*.

Among the three verification states, $\mathcal{R}$ and $\mathcal{A}$ admit relatively direct interpretations: blocks in $\mathcal{R}$ are already rejected by the incoming verification, whereas blocks in $\mathcal{A}$ remain consistent with both their preceding context and subsequent progression. The ambiguity therefore concentrates in $\mathcal{M}$. For $i \in \mathcal{M}$, the incoming evidence accepts b_i, while the outgoing evidence indicates that the progression from b_i to b_{i+1} is inconsistent. This disagreement localizes the uncertainty to the neighborhood of the transition, but does not yet reveal whether the underlying error lies in b_i, in b_{i+1}, or in a finer reasoning step hidden by the current block decomposition. PRO therefore treats $\mathcal{M}$ as the set of unresolved progression regions and subjects only these regions to the granularity-refinement stage described next.

4.3 Mismatch-triggered Granularity Refinement

The mismatch set $\mathcal{M}$ identifies where forward-local verification becomes insufficient, but it still does not reveal which fine-grained inference actually breaks the progression. In natural-language reasoning, a single block often compresses multiple semantic or computational operations, so the true point of failure may be hidden inside the block or at the transition into the next one. As a result, a progression mismatch should not yet be treated as a final detector decision. Instead, it marks a structurally ambiguous region that requires more precise inspection before source attribution.

To resolve this ambiguity, PRO introduces a mismatch-triggered coarse-to-fine refinement stage. Crucially, this additional verification is not applied to the entire trace. It is activated only for $i \in \mathcal{M}$, so that the extra effort is concentrated on local regions where the progression signal remains unresolved.

Local progression window. For each $i \in \mathcal{M}$, we define the local progression window as $\mathcal{W}_i = (b_i, b_{i+1})$, which isolates the neighborhood in which the disagreement first appears. We then decompose this local window into finer reasoning units:

$$\widehat{\mathcal{W}}_i = \text{Refine}(\mathcal{W}_i) = (u_{i,1}, u_{i,2}, \ldots, u_{i,m_i}), \qquad m_i \geq 2. \tag{5}$$

Fine-grained re-evaluation. Once the ambiguous region has been exposed at a finer resolution, PRO re-evaluates whether the local progression still fails:

$$\widehat{V}_i = \text{Verify}_{\text{fine}}\big(\widehat{\mathcal{W}}_i; q\big) \in \{0, 1\}. \tag{6}$$

Here, $\widehat{V}_i = 1$ means that the coarse mismatch can be reconciled after refinement, whereas $\widehat{V}_i = 0$ means that the inconsistency persists even under finer inspection. This step therefore determines

whether the original disagreement reflects only a coarse decomposition artifact or a more robust local failure.

Refined detector state. These refined outcomes are then folded back into the detector stage. Specifically, the detector state is defined as: $D_i = 1$ for $i \in \mathcal{R}$ and for $i \in \mathcal{M}$ with $\widehat{V}_i = 0$, whereas $D_i = 0$ for $i \in \mathcal{A}$ and for $i \in \mathcal{M}$ with $\widehat{V}_i = 1$.

This definition preserves the direct interpretations of $\mathcal{R}$ and $\mathcal{A}$, while using refinement only to adjudicate the ambiguous mismatch cases. The earliest position at which an inconsistency survives refinement is:

$$\tau_{\mathrm{det}}(q, \mathcal{B}) = \min\left(\{i \mid D_i = 1\} \cup \{\infty\}\right). \tag{7}$$

In this way, refinement converts a coarse progression mismatch into a more reliable detector signal. We emphasize that τ_{det} is still an *exposure-oriented* quantity: it marks the earliest robust point at which the trajectory can no longer be locally reconciled after ambiguity has been reduced, rather than directly asserting the first erroneous block. This refined exposure signal is therefore what the next stage uses as an anchor for source attribution.

4.4 Origin-oriented Candidate Attribution

The refined detector output identifies where the failure becomes robustly observable, but observable failure is still not the same as causal origin. A downstream block may be the first point that remains irreconcilable after refinement, even though the actual error was introduced earlier and only propagated to that position. The role of the final stage is therefore to reason backward from this refined exposure signal and determine which earlier block best explains the observed failure.

Candidate shortlisting. Let the refined detector return $j_{\mathrm{det}} \in \{1, \ldots, n\}$; if no inconsistency survives refinement, PRO predicts that no localizable error has been detected. Otherwise, j_{det} provides an anchor for source localization. In principle, the error origin should not occur later than the point where the failure becomes observable. In the implemented pipeline, we therefore use j_{det} to condition candidate attribution while still allowing the proposal stage to score the full trace, so that the detector output is treated as a strong exposure-aware hypothesis rather than a hard truncation rule.

Rather than treating every proposed block equally, PRO first obtains an ordered proposal list over the full trace as:

$$\pi\big(q, \mathcal{B}\big) = (c_1, \ldots, c_m), \qquad m \le k, \tag{8}$$

Here, $\pi(q, \mathcal{B})$ denotes the ordered output of an LLM-based source proposal function over the full trace. The detector position is then retained in the shortlist so that the exposed failure itself remains available as a competing hypothesis even when the proposal model would otherwise omit it. We write the resulting candidate set as:

$$\pi^{\star}_{k}\big(q, \mathcal{B}; j_{\mathrm{det}}\big) = (c^{\star}_1, \ldots, c^{\star}_{m^{\star}}), \qquad m^{\star} \le k. \tag{9}$$

This shortlisting step narrows the attribution problem from the full trajectory to a small set of plausible source hypotheses, while keeping the refined detector position available as a fallback explanation.

Intervention-based source evidence. Observation alone may still favor a downstream manifestation, particularly when several preceding blocks remain locally plausible under the corrupted trajectory. To distinguish these competing hypotheses, PRO introduces a lightweight intervention for each shortlisted candidate. For $c^{\star}_r$, the corresponding block is locally repaired under the same preceding context:

$$\widetilde{b}_{c^{\star}_r} = \mathrm{Rewrite}\big(q, \mathcal{P}_{c^{\star}_r - 1}, b_{c^{\star}_r}\big). \tag{10}$$

The repaired block is then re-evaluated using the same incoming and outgoing evidence. Let $\widetilde{V}^{(r)}_{\mathrm{in}}$ and $\widetilde{V}^{(r)}_{\mathrm{out}}$ denote the resulting verification signals. We define:

$$S_r = \mathbb{I}\left[\mathrm{Fuse}\left(\widetilde{V}^{(r)}_{\mathrm{in}}, \widetilde{V}^{(r)}_{\mathrm{out}}\right) = \mathrm{CORRECT}\right], \tag{11}$$

with an optional local refinement when the fused verdict remains ambiguous.

The purpose of S_r is not to independently certify a candidate as the first error. Instead, it measures whether repairing that candidate removes the local inconsistency associated with the observed progression failure. A candidate whose repair restores consistency receives stronger evidence of lying upstream of the exposure than one whose repair leaves the failure unchanged. In this way, rewrite-based verification functions as source-oriented intervention evidence rather than as a standalone correction heuristic.

Source attribution. Once the detector anchor and intervention-based evidence have been assembled, PRO makes the final attribution decision by jointly considering candidate chronology, detector position, trajectory context, and intervention outcomes:

$$j_{\mathrm{src}} = \mathrm{Rerank}\left(q, \mathcal{B}, j_{\mathrm{det}}, \{(c_r^\star, r, S_r)\}_{r=1}^{m^\star}\right). \tag{12}$$

The localized first-error prediction can be formulated as:

$$f(q, \mathcal{B}) = \begin{cases} j_{\mathrm{src}}, & j_{\mathrm{src}} \in \{c_1^\star, \ldots, c_{m^\star}^\star\}, \\ j_{\mathrm{det}}, & \text{otherwise.} \end{cases} \tag{13}$$

This final stage completes the progression-aware pipeline introduced above. PRO first identifies where the trajectory becomes robustly inconsistent, then uses that exposure signal to organize candidate hypotheses, test them through intervention, and trace the failure back to its most plausible source. By explicitly separating exposure detection from source attribution, PRO can use downstream consequences as retrospective evidence without assuming that the first rejected block is necessarily the first erroneous one.

5 Experiments

5.1 Experimental Setting

We compare PRO with three baselines: Holistic Verification, GoV (Fang et al., 2026), and PARC (Mukherjee et al., 2025). Holistic Verification evaluates the full reasoning trace in a single pass, GoV represents forward-only decomposition-based verification, and PARC provides a premise-augmented graph baseline. All experiments use deterministic decoding at temperature 0, and PRO uses a detector-conditioned shortlist of $k = 5$. For ProcessBench, Holistic Verification follows the official leaderboard result, while the other methods are evaluated under matched backbone settings.

We evaluate PRO on three tasks with complementary reasoning structures. For open-form reasoning, we use the GSM8K subset of ProcessBench (Zheng et al., 2025), containing 400 traces with human-annotated first-error positions. For medical reasoning, we construct a 450-trace MedReason benchmark from publicly available traces (Wu et al., 2025) following the MedPRMBench corruption protocol (Wu et al., 2026). For structured arithmetic reasoning, we use Number Triangle Summation with depths ($N \in \{2, 4, 6, 8\}$) and a 50% corruption rate. Metric are detailed in Appendix E.2.

5.2 Open-Form Reasoning Experiment

We first evaluate PRO under open-form reasoning using the GSM8K subset of ProcessBench. This setting contains naturally written multi-step solutions whose intermediate steps are not constrained by a fixed reasoning template, making the correspondence between the first erroneous step and the first observable failure less explicit. This experiment is designed to assess whether PRO can reliably localize the error source when reasoning errors may propagate through semantically plausible intermediate steps before becoming observable.

Table 1: ProcessBench. Performance comparison on ProcessBench (%).

Method	Metric	Qwen2.5-32B -Instruct	Qwen2.5-72B -Instruct	DeepSeek -V3.2	Llama3.3-70B -Instruct
Holistic Verification	Correct Acc	98.45	93.26	79.27	94.30
	Error Acc	45.89	65.70	80.68	68.60
PARC	Correct Acc	83.94	75.13	82.90	89.12
	Error Acc	57.97	68.60	77.78	63.77
GoV	Correct Acc	94.82	97.41	95.34	93.26
	Error Acc	68.12	71.01	79.71	68.60
PRO	Correct Acc	94.30	90.67	95.34	92.75
	Error Acc	68.12	76.81	80.68	70.53

Table 1 reports the quantitative comparison across different verification methods and backbone models. PRO generally achieves stronger first-error localization while maintaining competitive performance on correct traces, indicating that progression-aware verification provides a more reliable basis for source attribution than holistic or purely forward-local verification. Figure 2 further illustrates the overall performance trend across backbones. The visual comparison shows that the advantage of PRO is not tied to a particular model, but persists across different backbone choices. This consistency suggests that the improvement mainly comes from the verification strategy itself by explicitly separating error exposure from error origin, PRO is better able to recover the source of failures in loosely structured reasoning traces.

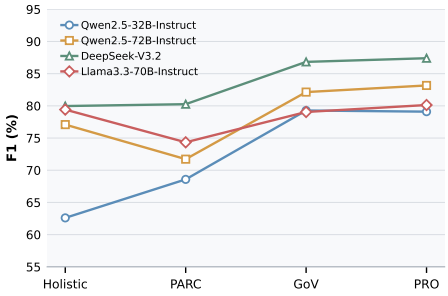

Figure 2: F1 trends across methods. Each line denotes one backbone.

5.3 Medical Reasoning Experiment

We evaluate PRO on medical reasoning to examine whether its source-localization capability extends beyond general-purpose reasoning to a domain with stronger semantic and clinical constraints. Compared with open-form arithmetic reasoning, medical traces introduce an additional challenge: intermediate statements can remain medically plausible in isolation even when an earlier decision has already placed the reasoning on an incorrect diagnostic path. As a result, the first locally suspicious step may be separated even further from the decision that actually introduces the error. The MedReason setting therefore tests whether PRO can preserve accurate first-error localization when error propagation occurs within domain-consistent but clinically consequential reasoning trajectories.

Table 2: MedReason. Performance comparison on MedReason (%).

Method	Metric	Qwen2.5-32B -Instruct	Qwen2.5-72B -Instruct	DeepSeek -V3.2	Llama3.3-70B -Instruct
Holistic Verification	Correct Acc	98.67	84.89	33.33	95.56
	Error Acc	17.78	27.11	44.89	11.56
PARC	Correct Acc	88.00	78.22	68.44	89.33
	Error Acc	25.33	35.56	23.56	17.78
GoV	Correct Acc	81.78	80.44	59.11	88.44
	Error Acc	27.56	28.89	33.78	12.44
PRO	Correct Acc	71.11	72.44	48.89	86.67
	Error Acc	40.00	38.22	42.22	27.11

Table 2 reports the quantitative comparison on MedReason. PRO shows consistently stronger first-error localization across the evaluated backbones, with the improvement primarily reflected in its ability to identify erroneous traces while retaining reliable recognition of correct reasoning. This suggests that progression-aware verification is beneficial in medical reasoning, where locally plausible statements can obscure an earlier diagnostic error and make forward-local judgments less reliable for source attribution.

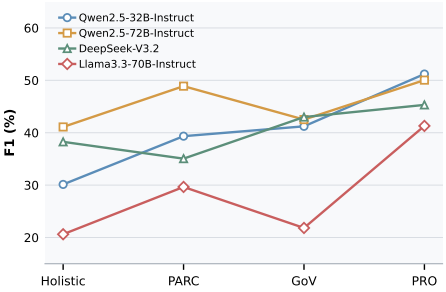

Figure 3: MedReason F1 trends. Each line denotes one backbone.

Figure 3 further shows the overall performance trend across different backbones. PRO maintains a clear and stable advantage despite changes in the underlying model, indicating that the improvement is not specific to a particular backbone. This trend supports the central premise of PRO, under domain-constrained reasoning, downstream clinical coherence does not necessarily identify the true error source, and explicitly reasoning over error progression helps recover the earlier step that initiates the failure.

5.4 Cross-Backbone Statistics

To assess whether the improvement of PRO is robust to the choice of backbone, we further aggregate its performance gains across the four evaluated models. Specifically, for each benchmark and baseline, we compute the F1 improvement of PRO under each backbone and then average these gains across backbones.

Figure 4 summarizes the resulting average gains on ProcessBench and MedReason. PRO maintains positive improvements over all baselines across both benchmarks, indicating that its advantage is not driven by a particular backbone or isolated experimental configuration. The improvement is especially more pronounced on MedReason, where clinically plausible intermediate reasoning makes the separation between error exposure and error origin more challenging.

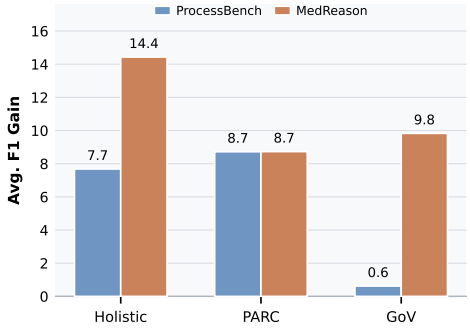

Figure 4: Cross-backbone summary statistics.

5.5 Structured Arithmetic Reasoning

Table 3: The results on Number Triangle Summation (%).

Problem Size	Llama3.3-70B-Instruct	
	GoV	PRO
$N = 2$	**99.40**	98.37
$N = 4$	90.59	**94.13**
$N = 6$	72.30	**79.82**
$N = 8$	46.62	**56.54**

We further evaluate PRO on Number Triangle Summation. Unlike open-form and medical reasoning, this task has an explicit dependency structure, while the reasoning depth can be systematically varied through the problem size N. Table 3 reports the results across different reasoning depths. When the reasoning chain is short, PRO and GoV exhibit comparable performance, as an introduced error has limited opportunity to propagate before becoming observable. As the reasoning depth increases, PRO shows an increasingly clearer advantage. This trend suggests that the benefit of progression-aware localization becomes more pronounced as the distance between error origin and error exposure grows. The controlled results therefore complement the observations on ProcessBench and MedReason, showing that PRO's advantage is closely associated with error propagation rather than with a particular reasoning domain or semantic structure.

5.6 Ablation Study

We further examine the contribution of each component of PRO on ProcessBench using Qwen2.5-32B-Instruct. Table 4 compares the full model with variants of PRO. The results show that all three components contribute to first-error localization. Removing outgoing evidence weakens the ability to distinguish locally plausible blocks from progression-inconsistent ones, while removing refinement makes ambiguous detector outputs harder to resolve. The largest degradation occurs without top-k reranking, indicating that detector-conditioned source attribution is particularly important for converting exposed failures into accurate error origins.

Table 4: The ablation study of PRO.

Backward	Refinement	Top-k Rerank	Err. Acc.	F1
×	×	×	56.28	68.64
×	✓	✓	58.94	69.28
✓	×	✓	50.72	64.51
✓	✓	×	44.93	57.78
✓	✓	✓	**68.12**	**79.10**

6 Conclusion

In this paper, we identify a key limitation of sequential reasoning verification. Under error propagation, the first observable failure may not be the step that actually introduces the error. To address this *source–exposure* gap, we propose a training-free framework PRO that models how verification evidence evolves along the reasoning trajectory. PRO combines progression-aware evidence, selective refinement, and source-oriented attribution to trace exposed failures back to their likely origin. Its effectiveness comes from separating exposure detection from source localization and using downstream consequences as retrospective evidence for earlier errors. Experiments across diverse reasoning settings validate this design. More broadly, our results suggest that reliable reasoning verification should model error propagation rather than rely solely on local stepwise judgments. Future work will extend this work to explor longer reasoning traces.

AI USE STATEMENT

We used generative AI tools only for limited writing assistance during manuscript preparation, including language polishing, grammar correction, and improvements to stylistic clarity. We did not use generative AI tools for method design, experimental design, result generation, or scientific claim validation. All AI-assisted content was reviewed by the authors. The authors verified all technical content, method design, experimental settings, results, and final wording decisions, and take full responsibility for the final contents of this paper.

REFERENCES

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 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. doi: 10.1609/AAAI.V38I16.29720.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Łukasz 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.

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

Jiwei Fang, Bin Zhang, Changwei Wang, Jin Wan, and Zhiwei Xu. Graph of verification: Structured verification of LLM reasoning with directed acyclic graphs. In *Proceedings of the 40th AAAI Conference on Artificial Intelligence*, pp. 30665–30672, 2026. doi: 10.1609/AAAI.V40I36.40322.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Sx038qxjek.

Yancheng He, Shilong Li, Jiaheng Liu, Weixun Wang, Xingyuan Bu, Ge Zhang, Zhongyuan Peng, Zhaoxiang Zhang, Zhicheng Zheng, Wenbo Su, and Bo Zheng. Can large language models detect errors in long chain-of-thought reasoning? In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 18468–18489, 2025. doi: 10.18653/v1/2025.acl-long.905.

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

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

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.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in language models with automated process supervision. *arXiv preprint arXiv:2406.06592*, 2024.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, 2023.

Sagnik Mukherjee, Abhinav Chinta, Takyoung Kim, Tarun Anoop Sharma, and Dilek Hakkani-Tur. Premise-augmented reasoning chains improve error identification in math reasoning with LLMs. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 45109–45128, 2025. URL https://proceedings.mlr.press/v267/mukherjee25a.html.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Guangya Wan, Yuqi Wu, Hao Wang, Shengming Zhao, Jie Chen, and Sheng Li. Derailer-rerailer: Adaptive verification for efficient and reliable language model reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025. doi: 10.18653/v1/2025.findings-acl.18.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, 2024.

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*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. Large language models are better reasoners with self-verification. *arXiv preprint arXiv:2212.09561*, 2023.

Juncheng Wu, Wenlong Deng, Xingxuan Li, Sheng Liu, Taomian Mi, Yifan Peng, Ziyang Xu, Yi Liu, Hyunjin Cho, Chang-In Choi, Yihan Cao, Hui Ren, Xiang Li, Xiaoxiao Li, and Yuyin Zhou. Medreason: Eliciting factual medical reasoning steps in llms via knowledge graphs. *arXiv preprint arXiv:2504.00993*, 2025. doi: 10.48550/arXiv.2504.00993. URL https://arxiv.org/abs/2504.00993.

Lingyan Wu, Xiang Zheng, Weiqi Zhai, Wei Wang, Xuan Ren, Zifan Zhang, Hu Wei, and Bing Zhao. Medprmbench: A fine-grained benchmark for process reward models in medical reasoning. *arXiv preprint arXiv:2604.17282*, 2026. doi: 10.48550/arXiv.2604.17282. URL https://arxiv.org/abs/2604.17282.

Zhenyu Wu, Qingkai Zeng, Zhihan Zhang, Zhaoxuan Tan, Chao Shen, and Meng Jiang. Large language models can self-correct with key condition verification. *arXiv preprint arXiv:2405.14092*, 2024.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Processbench: Identifying process errors in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1009–1024, 2025.

APPENDIX

This appendix provides supplementary material for the main paper. Appendix A extends the related-work discussion beyond the compact summary in the main text. Appendix B introduces the implementation-side preliminaries repeatedly used in the supplementary sections. Appendix C

gives the theoretical proofs. Appendix D summarizes the main steps of PRO from the detector to the source-localization stage. Appendix E records the experimental setting and reproducibility protocol. Appendix F presents an additional qualitative case study.

A Related Work

A.1 Reasoning Verification

Reasoning verification is commonly instantiated either as holistic verification over the full chain or as decomposition-based verification over intermediate units (Cobbe et al., 2021; Uesato et al., 2022; Fang et al., 2026). Holistic verifiers provide a single global judgment over the complete trace and remain a strong baseline in many mathematical reasoning settings, especially when the main objective is to determine whether a full solution is acceptable. Their limitations become more visible when the trajectory is long, when multiple local dependencies interact, or when the task requires localized error attribution rather than a single final verdict.

Decomposition-based methods instead expose intermediate units to the verifier, trading simplicity for finer-grained evidence and stronger interpretability. Existing approaches differ in what they treat as the basic unit of verification, ranging from individual steps to adaptive blocks that follow task structure. GoV is representative of the latter direction because it treats verification granularity as a structural choice rather than a fixed design constant (Fang et al., 2026). Recent benchmarks such as ProcessBench and DeltaBench make this distinction especially salient by showing that earliest-error localization remains difficult even for strong process verifiers (Zheng et al., 2025; He et al., 2025).

A.2 Process Supervision

Process supervision provides learning signals at the level of intermediate reasoning steps rather than only final outcomes (Uesato et al., 2022; Lightman et al., 2023). This line of work has been influential because it shows that reasoning quality cannot always be recovered from outcome supervision alone: step-level labels often reveal errors that are invisible from final answers. PRM-style supervision and its lower-cost extensions, such as Math-Shepherd and OmegaPRM, further demonstrate that intermediate supervision can be made more scalable through semi-automatic annotation or teacher-assisted labeling (Wang et al., 2024; Luo et al., 2024).

The broader literature in this area therefore studies not only whether step-level information matters, but also how such information should be collected, represented, and used. Most existing approaches focus on learned critics, reward models, or annotation pipelines for constructing step-level labels. As a result, process supervision has developed into both a methodological direction for training reasoning systems and an evaluation lens for understanding where intermediate errors arise.

A.3 Self-Correction and Test-Time Verification

Test-time verification and self-correction improve reasoning by inserting critique, revision, or verification during inference (Madaan et al., 2023; Dhuliawala et al., 2024; Wu et al., 2024; Wan et al., 2025). These methods mainly target answer improvement or trajectory repair, and they highlight the broader value of inference-time feedback without retraining the base model. Some methods emphasize iterative refinement, while others explicitly separate generation from verification so that the model can revise its own reasoning under external checks.

This literature is also closely connected to recent discussions about the limits of intrinsic self-correction. Huang et al. (2023) show that models often fail to reliably repair their own reasoning without sufficiently informative external signals. This observation has motivated continued interest in external critics, verification-guided refinement, and hybrid test-time pipelines that combine detection, critique, and revision.

B PRELIMINARIES

B.1 PROBLEM SETTING

This paper studies *first-error localization* for multi-step reasoning traces. Given a problem instance q and a reasoning trace $\mathcal{B} = \{b_1, \ldots, b_n\}$, the verifier must return the earliest incorrect block index, or -1 if the full trace is valid. Unlike final-answer verification, this setting requires the verifier to reason about intermediate dependencies inside the trajectory rather than only judging the final output. The same formulation is used across all three datasets in our paper, although the trace structure differs across arithmetic programs, free-form mathematical derivations, and medical reasoning chains.

Verification Setting: The central difficulty is that the first *exposed* inconsistency is not always the true *source* error. A downstream block may be the first place where failure becomes visible, even though the underlying mistake originates in an earlier block. As a result, earliest-error localization cannot be reduced to simply finding the first block that looks suspicious under local checking.

Detector and Candidate Space: To handle this mismatch, the appendix repeatedly distinguishes detector-stage evidence from source-localization evidence. Concretely, the detector returns an index $j_{\mathrm{det}} \in \{-1, 1, \ldots, n\}$, where -1 means that no suspicious block survives block-level screening. The subsequent top-k stage begins with a proposal list over the full trace, but the final shortlist is detector-conditioned: it always contains j_{det} when $j_{\mathrm{det}} \neq -1$, even if the ranker itself would otherwise omit that block.

Implementation Conventions: We also adopt two conventions that are used throughout the appendix. For the final block b_n, we set $V_{\mathrm{out}}(b_n) = 1$, so that the terminal block is not marked uncertain merely because no successor exists. In addition, we follow a fail-closed protocol: empty responses, malformed verdicts, and API-side failures are not silently promoted to correct judgments, but are treated as uncertain or failed verification outcomes.

B.2 GOALS

The purpose of PRO is not merely to detect that a reasoning trace fails, but to improve the reliability of earliest-source localization under black-box, training-free verification. To make this objective concrete, the method is designed around the following goals.

- **Localization Fidelity.** The verifier should recover the earliest source error rather than only the first downstream block that visibly breaks. This requires the pipeline to distinguish causal origin from propagated inconsistency, rather than treating every exposed downstream failure as equally informative.
- **Structural Robustness.** The method should remain effective when trace length, block granularity, and local dependency structure vary substantially across datasets. In other words, the verifier should not depend on a single fixed notion of what a "natural" reasoning unit looks like, since arithmetic traces, free-form derivations, and medical reasoning chains expose very different local structures.
- **Training-Free Deployability.** The full pipeline should work without additional process-supervision labels, learned reward models, or task-specific verifier training. This constraint is important because the intended setting is black-box verification rather than supervised critic construction, and therefore the method must rely on pipeline design rather than extra training signals.

These goals motivate the notation used in the remainder of the appendix: later proofs and analyses repeatedly rely on the distinction between incoming-side signals, outgoing-side mismatch, detector-conditioned candidate construction, and rewrite-supported reranking.

B.3 CHALLENGES

The above goals are difficult to satisfy simultaneously because earliest-source localization is not simply a stronger form of ordinary correctness checking. The core difficulty lies in identifying the causal origin of failure rather than merely the first place where failure becomes visible, and doing so

under reasoning structures whose local granularity may itself be unstable. In our setting, the main challenges are therefore structural rather than purely operational.

Challenge I: Exposure-Source Mismatch. In many error traces, the earliest block that looks suspicious is not the true origin of failure. This creates a systematic gap between detection and source attribution, because local verification evidence may first become visible only after the original mistake has already propagated. Conventional stepwise verification is therefore biased toward the first exposed downstream inconsistency, even when that inconsistency is merely a consequence of an earlier hidden mistake. The challenge is not only to flag that something has gone wrong, but to separate propagated failure from causal origin without assuming access to gold intermediate labels.

Challenge II: Structural Ambiguity Under Variable Granularity. Even after a suspicious region is exposed, source localization remains difficult because the same local mismatch may admit multiple explanations. A failed bridge can be caused by the current block, by its immediate continuation, or by a coarser inconsistency inherited from an earlier region. Fine-grained verification helps expose local arithmetic or logical faults, but overly small units may discard the context needed for causal attribution; coarser units preserve context, but they blur the precise location of error. A practical method must therefore refine granularity only where the structure becomes ambiguous, while preserving enough context to decide which candidate block is genuinely source-like.

C Theoretical Proof

The theoretical role of PRO is to formalize why forward-only rejection is insufficient for detector-stage structural exposure, and why outgoing-side structure must be incorporated before source localization can be carried out reliably. To support this claim, we prove two complementary statements: the structural-exposure index used by the detector can strictly precede the ordinary forward rejection index, and incoming-side signals alone are insufficient for universally correct recovery of that detector-stage target.

C.1 Forward Rejection Versus Structural Exposure

We first compare the earliest index returned by ordinary forward rejection with the structural-exposure index used by PRO. The claim is that the detector-stage index can never occur later than the forward rejection index, and can in fact be strictly earlier when outgoing-side mismatch exposes a suspicious bridge before any incoming-side rejection occurs.

Proof. **Candidate-set inclusion.** To show that structural exposure cannot occur later than forward rejection, we compare the two minimized index sets, which can be written as:

$$\tau_{\mathrm{seq}}(q, \mathcal{B}) = \min\big(\{\, i \mid I_i = 0 \,\} \cup \{\infty\}\big), \tag{A1}$$

and

$$\tau_{\star}(q, \mathcal{B}) = \min\big(\{\, i \mid I_i = 0 \,\} \cup \mathcal{M}(q, \mathcal{B}) \cup \{\infty\}\big), \tag{A2}$$

where $\mathcal{M}(q, \mathcal{B}) = \{\, i < n \mid I_i = 1,\ O_i = 0 \,\}$. The key set inclusion can be expressed as:

$$\{\, i \mid I_i = 0 \,\} \cup \{\infty\} \subseteq \{\, i \mid I_i = 0 \,\} \cup \mathcal{M}(q, \mathcal{B}) \cup \{\infty\}. \tag{A3}$$

Therefore, the desired ordering relation can be formulated as:

$$\tau_{\star}(q, \mathcal{B}) \leq \tau_{\mathrm{seq}}(q, \mathcal{B}). \tag{A4}$$

Strictness witness. To show that the inequality can be strict, it is enough to construct a trace in which all incoming-side tests pass, but an earlier outgoing-side mismatch exists. Consider any trace satisfying $I_i = 1$ for all i, and suppose there exists some $j < n$ such that $O_j = 0$. Then $j \in \mathcal{M}(q, \mathcal{B})$, so $\tau_{\star}(q, \mathcal{B}) \leq j < \infty$, while $\tau_{\mathrm{seq}}(q, \mathcal{B}) = \infty$. Hence the inequality is strict on this instance. □

C.2 Necessity of Outgoing-Side Information

We next show that two traces can share identical incoming-side signals while differing in their detector-stage structural-exposure index. This establishes that incoming-side evidence alone is not sufficient for universally correct recovery of that detector target.

Proof. **Matched incoming-side signals.** We construct two traces over the same problem instance q that share identical incoming-side signals but differ in outgoing-side structure. Let $\mathcal{B} = (b_1, \ldots, b_n)$ and $\mathcal{B}' = (b'_1, \ldots, b'_n)$ be two decomposed traces such that $I_k(\mathcal{B}) = I_k(\mathcal{B}') = 1$ for all k.

Different outgoing-side structure. Assume furthermore that $\mathcal{B}$ contains an outgoing-side mismatch at the first block, $O_1(\mathcal{B}) = 0$, whereas $\mathcal{B}'$ has no mismatch at all, with $O_k(\mathcal{B}') = 1$ for all $k < n$. Then $\mathcal{M}(q, \mathcal{B}) \ni 1$, whereas $\mathcal{M}(q, \mathcal{B}') = \varnothing$. Since all incoming-side signals are identical and always equal to 1, the difference between the two traces must come entirely from outgoing-side structure. The corresponding detector-stage structural-exposure indices can be expressed as:

$$\tau_\star(q, \mathcal{B}) = 1, \qquad \tau_\star(q, \mathcal{B}') = \infty. \tag{A5}$$

Therefore, their separation can be formulated as:

$$\tau_\star(q, \mathcal{B}) \neq \tau_\star(q, \mathcal{B}'). \tag{A6}$$

Implication for detector rules. Any detector rule that depends only on the incoming-side sequence $\{I_k\}$ would assign the same output to $\mathcal{B}$ and $\mathcal{B}'$, because those signals are identical by construction. Such a rule cannot be correct on both traces simultaneously. Hence outgoing-side information is necessary for universally correct recovery of $\tau_\star$. □

D THE MAIN STEPS OF PRO

PRO proceeds in three stages at test time. First, it performs block-level bidirectional checking using incoming admissibility and outgoing-side support against the immediate continuation. When the two signals disagree, PRO treats the case as a structural ambiguity and enters local refinement rather than issuing an immediate rejection. The theoretical pre-refinement exposure index is $\tau_\star$, while the refined detector output used by the implementation is τ_{det}.

For every suspicious region that survives detection, PRO uses the refined detector output as an anchor for candidate attribution. It first obtains an ordered proposal list over the full trace, then constructs a detector-conditioned top-k shortlist by retaining the detector position when necessary before truncating the list to length k. Each shortlisted candidate is locally rewritten and re-verified under the same verified prefix, and the resulting repair evidence is used for source reranking. If reranking does not return a valid candidate from the shortlist, PRO falls back to the original detector index.

Across datasets, PRO uses the same prompt family for forward verification, outgoing-side verification, edge refinement, candidate proposal, candidate rewriting, repair verification, and source reranking; only the verification unit definition changes with the task. For the terminal block b_n, we set $V_{\mathrm{out}}(b_n) = 1$, so that the absence of a successor does not downgrade the final block. Algorithm 1 summarizes the complete test-time procedure.

E EXPERIMENTAL SETTING

E.1 INFERENCE AND DATA CONFIGURATION

All compared methods use the same backbone within each setting. The main evaluation runs use deterministic decoding at temperature 0, while any retry behavior follows the released task-specific scripts. PRO uses a detector-conditioned shortlist of $k = 5$. ProcessBench is evaluated on its complete GSM8K, MATH, and OlympiadBench test subsets. The MedReason evaluation set contains 450 traces, comprising 225 unmodified traces and 225 corrupted counterparts balanced across nine source partitions. The injected error position is used as the first-error label, and the same evaluation set is used for every compared backbone.

E.2 METRICS

In this paper, F1 always denotes exact first-error localization F1. When dataset annotations permit, we additionally report Correct Accuracy and Error Accuracy. These definitions are used throughout

Algorithm 1 The main steps of PRO

Require: $q, \mathcal{B} = (b_1, \ldots, b_n)$, shortlist size k
Ensure: $\hat{f}(q, \mathcal{B}) \in \{-1, 1, \ldots, n\}$

$j_{\text{det}} \leftarrow \infty$
for $i = 1$ to n **do**
  $I_i \leftarrow V_{\text{in}}(q, \mathcal{P}_{i-1}, b_i)$
  $O_i \leftarrow \mathbf{1}[i = n] + \mathbf{1}[i < n] \cdot V_{\text{out}}(q, b_i, b_{i+1})$
  $r_i \leftarrow \varnothing$
  if $I_i = 1 \wedge O_i = 0$ **then**
    $r_i \leftarrow \text{Refine}(q, \mathcal{P}_{i-1}, b_i, b_{i+1})$
  end if
  if $\text{Detect}(I_i, O_i, r_i) = 1$ **then**
    $j_{\text{det}} \leftarrow i$; **break**
  end if
end for
if $j_{\text{det}} = \infty$ **then**
  return -1
end if
$(c_1, \ldots, c_m) \leftarrow \text{Propose}(q, \mathcal{B})$
$(c_1^\star, \ldots, c_{m^\star}^\star) \leftarrow \text{Shortlist}(q, \mathcal{B}, (c_1, \ldots, c_m), k, j_{\text{det}})$
for $r = 1$ to $m^\star$ **do**
  $j \leftarrow c_r^\star$
  $\tilde{b}_r \leftarrow \text{Rewrite}(q, \mathcal{P}_{j-1}, b_j)$
  $e_r \leftarrow \text{VerifyRepair}(q, \mathcal{B}, j, \mathcal{P}_{j-1}, \tilde{b}_r)$
end for
$j^\star \leftarrow \text{Rerank}(q, (c_1^\star, \ldots, c_{m^\star}^\star), \{(r, e_r)\}_{r=1}^{m^\star}, j_{\text{det}})$
if $j^\star \in \{c_1^\star, \ldots, c_{m^\star}^\star\}$ **then**
  return $j^\star$
else
  return j_{det}
end if

the main text and appendix because they directly reflect whether the verifier recovers the true source step rather than merely exposing that some downstream failure exists.

• **Correct Accuracy:** The proportion of fully correct reasoning traces that are correctly judged as error-free.

• **Error Accuracy:** The proportion of erroneous reasoning traces for which the method successfully identifies the first incorrect step.

• **F1:** The harmonic mean of Correct Accuracy and Error Accuracy, used as the primary metric for overall first-error localization performance.

F Qualitative Case Study

To understand how the gain appears at the trajectory level, we inspect a representative MedReason example in which GoV fails to recover the annotated source step, while PRO localizes it correctly after candidate-conditioned rewriting and reranking.

The selected example asks for the primary mechanism of islet-cell injury in a 10-year-old boy with recent-onset diabetes requiring daily insulin therapy, whose clinical picture strongly suggests Type 1 diabetes mellitus. In the corrupted trajectory, the downstream intermediate steps (Steps 3 and 4) remain locally plausible: they correctly introduce HLA susceptibility, antigen presentation, CD8+ T cells and inflammatory cytokines as supporting immune factors for T1DM. The actual corruption appears early in Step 2, in the form of an unsupported early commitment to a causal claim. This error is subtle and propagates through subsequent reasoning; downstream steps remain coherent until Step 5, where the factual mistake ("furosemide-producing beta cells") becomes explicit.

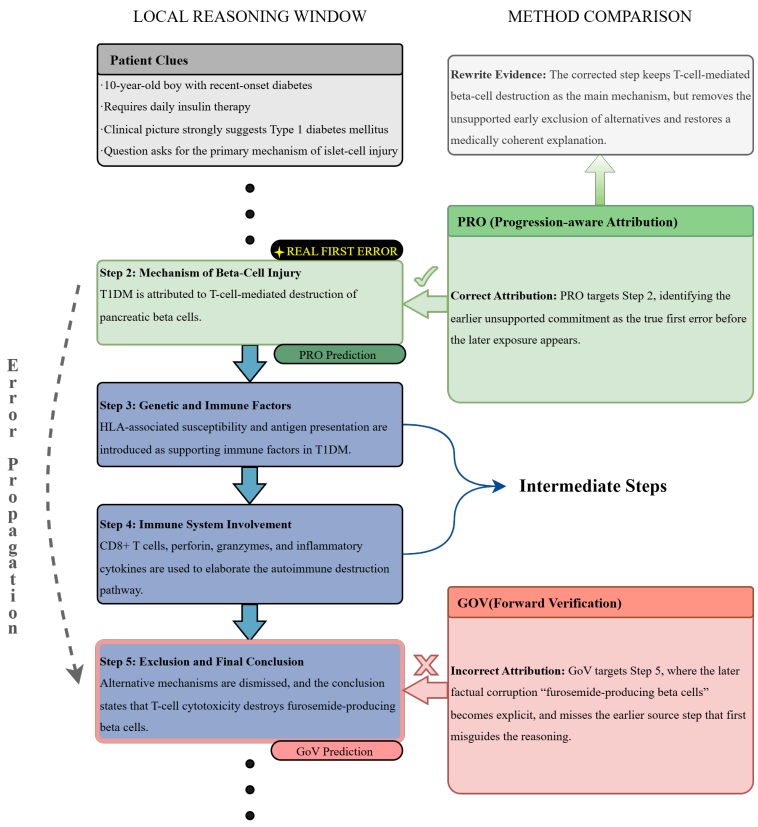

Figure A1: Representative MedReason case study. Type 1 diabetes reasoning trajectory. The true first error is at Step 2. GoV misattributes the failure to Step 5 where corruption becomes explicit, while PRO correctly identifies the early source error.

GoV does not isolate this source step reliably, since the intermediate statements about genetic susceptibility and immune mediators still look acceptable under the corrupted prefix. PRO instead revisits the suspicious region with candidate-conditioned rewriting, tests whether a revised version restores consistency with both the preceding clinical context and the final conclusion, and then reranks the surviving candidates by source plausibility. In this case, PRO identifies Step 2 as the first erroneous block and recovers the true source location. This example illustrates why progression-aware attribution helps in medically constrained traces: locally plausible intermediate reasoning steps can still be built upon an earlier unjustified claim, and correcting that initial flawed assembly point matters more than simply flagging the first downstream factual mismatch.