Conformal Certification of Reasoning Trace Prefixes

Anonymous ACL submission

Abstract

Language model reasoning traces are rarely all-or-nothing; they frequently contain valid intermediate steps before a critical error occurs. Existing uncertainty quantification methods typically certify final answers or entire responses, failing to provide statistical guarantees for the proportion of a sequential trace that can be safely retained. To address this, we introduce CROP (Conformal Reasoning Output Prefixes), a verifier-agnostic calibration procedure for clean-prefix certification. Given any step-level risk proxy, CROP selects a calibrated threshold and returns the longest contiguous prefix whose step risk proxies remain below it, routing the uncertified suffix for downstream review or repair. Assuming exchangeability, CROP rigorously controls the marginal probability that the returned prefix contains an annotated error. Across six process-labeled reasoning datasets, we demonstrate that standard step-level metrics such as AUROC do not fully capture prefix utility, suggesting verifiers should instead be evaluated by certified prefix length. Furthermore, CROP balances over- and under-withholding, improving downstream repair accuracy by preserving valid intermediate reasoning while discarding misleading suffixes. Ultimately, this work positions prefix certification as a rigorous, practical bridge between process supervision, abstention, and repair.

1 Introduction

Modern language models (LMs) solve complex problems by generating reasoning traces, or "chains-of-thought" (Wei et al., 2022). When an LM yields an incorrect final answer, its trace is rarely entirely wrong; valid early reasoning is typically followed by a critical mistake due to error propagation or context loss (Figure 1). Traditional uncertainty quantification (UQ) compresses this structure into a binary accept/reject decision, offering no guidance on which steps to trust or route for repair. Alternatively, step-level UQ signals ranging from likelihood statistics (Lin et al., 2024) to learned process verifiers (Lightman et al., 2024; Park et al., 2025) produce uncalibrated values that lack formal statistical guarantees when deployed.

Conformal prediction (CP) offers a robust statistical framework for converting arbitrary model-derived scores into rigorous guarantees. However, recent adaptations of CP for LMs (Campos et al., 2024) typically certify the final label (Kumar et al., 2023), candidate answer sets (Quach et al., 2024), factual claims (Mohri and Hashimoto, 2024), or the entire response (Yadkori et al., 2024). These choices do not align with process-level reasoning: certifying an entire trace is overly conservative, as a single late mistake invalidates earlier valid work, while treating claims independently ignores the sequential structure required to establish a contiguous trust boundary. Consequently, existing methods cannot directly provide *clean-prefix certification*: the selection of a contiguous, usable trace prefix that rigorously controls the probability of containing an error.

To bridge this gap, we introduce *Conformal Reasoning Output Prefixes* (CROP), a verifier-agnostic calibration procedure that extracts a contiguous prefix of reasoning steps while controlling the marginal probability that it contains an annotated error (Figure 1). Given a reasoning instance $X = (q, \tau)$ with question q and trace $\tau = (\tau_1, \ldots, \tau_T)$, CROP assigns each step a risk proxy $R_t = s_t(X)$. This fixed step-level proxy function, $s_t(\cdot)$, can be any signal such as a process reward model, learned detector, or likelihood statistic where larger values indicate higher estimated error risk. Using held-out labeled reasoning instances and a target risk level α, CROP returns the longest contiguous prefix $\tau_{1:m}$ whose step risks remain below a calibrated threshold. One may truncate and route the uncertified suffix $\tau_{m+1:T}$ for downstream review or repair, for example, using backtracking or self-correction

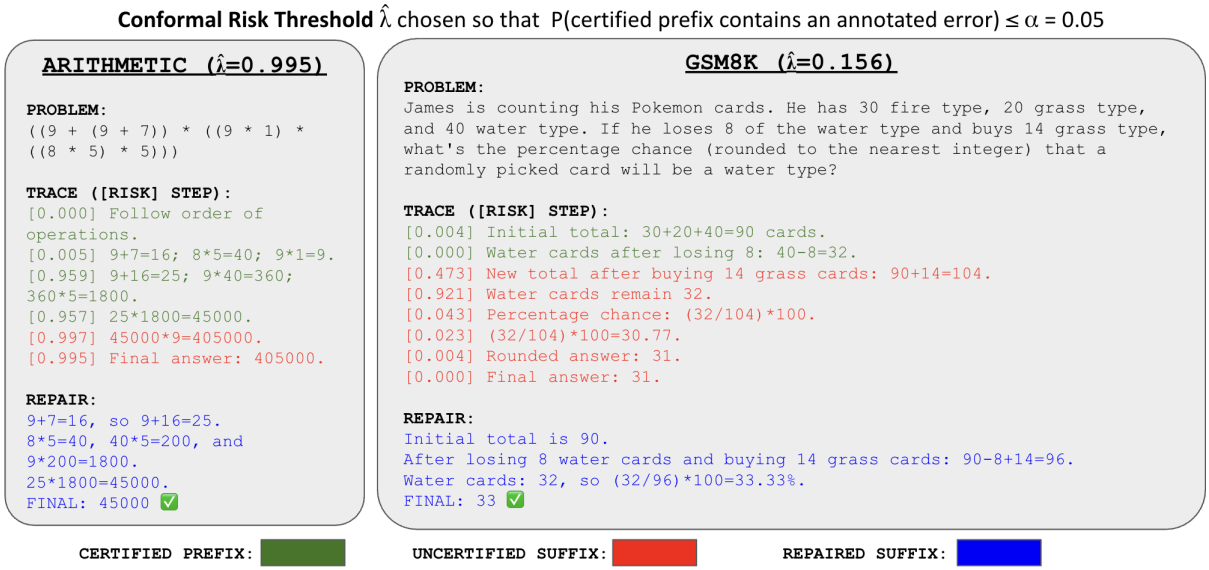

Figure 1: **CROP returns a calibrated prefix of a completed reasoning trace.** We show two example traces solving problems from the Arithmetic and GSM8K datasets. For each step in each reasoning instance, CROP computes a risk proxy, with larger values indicating higher estimated error risk. Using held-out calibration reasoning-instances, CROP selects a threshold that controls the marginal probability that the retained prefix contains an annotated erroneous step. At test time, CROP returns the longest contiguous prefix (green) whose risk proxies remain below the calibrated threshold. The remaining suffix (red) is left uncertified and can be routed to downstream review or repair (blue).

techniques (see Section 2).

We evaluated CROP across six process-labeled reasoning datasets: Arithmetic (Zhao et al., 2026), GSM8K (Zhao et al., 2026), ProcessBench (Zheng et al., 2025), Math-Shepherd (Wang et al., 2024a), PRMBench (Song et al., 2025), and PRM800K (Lightman et al., 2024). We demonstrate that standard step-level metrics such as AUROC fail to fully capture fixed-risk prefix utility, underscoring the necessity of our prefix-calibrated approach. On Arithmetic and GSM8K, CROP selects retained prefixes closer to the annotated clean prefixes than non-prefix interfaces. By preserving a calibrated prefix rather than accepting or discarding the whole trace, CROP better balances between over-withholding (discarding valid steps) and under-withholding (retaining unsafe steps), ultimately improving downstream repair accuracy over standard baselines in the majority of our benchmarks.

We summarize our contributions as follows: **1. Formalizing Clean-Prefix Certification:** We define clean-prefix certification for sequential reasoning traces and introduce CROP as a verifier-agnostic calibration layer that converts step-level risk proxies into a risk-controlled contiguous prefix. **2. Theoretical Guarantees:** We prove finite-sample marginal reasoning instance-level control of annotated prefix contamination under the assumption of exchangeability. **3. Empirical Effectiveness:** We demonstrate that AUROC is an insufficient predictor of prefix utility. CROP selects stopping points that stikes a better balance between over- and under-withholding, preserving valid intermediate reasoning and significantly enhancing downstream repair accuracy over uncalibrated UQ baselines.

2 Related Work

Uncertainty Quantification in LMs. Modern language models expose various uncertainty signals, but their raw values lack inherent calibration. Prior UQ methods leverage likelihoods (Kadavath et al., 2022; Lin et al., 2024), verbalized confidence (Lin et al., 2022; Tian et al., 2023), self-consistency (Wang et al., 2023), semantic uncertainty (Kuhn et al., 2023), sampling disagreement (Manakul et al., 2023), and hidden-state features (Azaria and Mitchell, 2023). While these signals rank relative risk, they require additional calibration layers to provide finite-sample reliability guarantees.

Reasoning Verifiers and Process Supervision. Beyond general generation, specialized verifiers

evaluate multi-step reasoning. These include process reward models (PRMs) (Lightman et al., 2024; Zhang et al., 2025b; Yang et al., 2024), scalable process supervision (Luo et al., 2024), attribution graphs (Zhao et al., 2026), and dependency-based step analysis (Mukherjee et al., 2025). CROP is verifier-agnostic, natively accepting risk proxies from any of these methods to determine a certified trust boundary.

Backtracking and Self-Correction. Recent work leverages backtracking to improve generation, reasoning, and safety. Methods either train models to emit explicit backtracking tokens or apply inference-time search to revise earlier decisions (Yang et al., 2025; Qin et al., 2025; Cai et al., 2025; Gandhi et al., 2025), or undo problematic continuations to recover from unsafe generations (Zhang et al., 2025a; Sel et al., 2025). These approaches modify the generation or repair policy itself. CROP is complementary to these methods: while CROP does not prescribe *how* a suffix should be regenerated, it provides backtracking and self-correction systems with a mathematically guaranteed, calibrated restart point.

Conformal Prediction in LMs. Conformal prediction (CP) and conformal risk control (CRC) convert arbitrary scores into prediction rules with marginal guarantees under exchangeability (Vovk et al., 2005; Fontana et al., 2023; Shafer and Vovk, 2008; Angelopoulos and Bates, 2023; Angelopoulos et al., 2024b,a). Recent adaptations for LM UQ (Campos et al., 2024) span multiple-choice QA (Kumar et al., 2023; Vishwakarma et al., 2025), open-ended generation (Quach et al., 2024; Ulmer et al., 2024; Wang et al., 2024b), and whole-trace abstention (Machcha et al., 2025; Yadkori et al., 2024). Closest to our problem are conformal factuality methods (Mohri and Hashimoto, 2024; Rubin-Toles et al., 2025; Cherian et al., 2024). However, these prior works typically certify final labels, answer sets, individual factual claims, dependency structures, or all-or-nothing abstention decisions. CROP diverges by certifying a contiguous trace prefix, preserving the original step order to maximize reusable context for downstream repair.

3 Problem Setup

Setup. Let q denote a question and $\tau = (\tau_1, \ldots, \tau_T)$ the corresponding generated reasoning trace, with step τ_t at index t. We denote the completed reasoning instance $X = (q, \tau)$. A reasoning trace prefix is the contiguous sequence $\tau_{1:m} = (\tau_1, \ldots, \tau_m)$, leaving the uncertified suffix $\tau_{m+1:T} = (\tau_{m+1}, \ldots, \tau_T)$. Dataset annotation assigns each step a binary error label $Y = (Y_1, \ldots, Y_T) \in \{0, 1\}^T$, where $Y_t = 1$ indicates an annotated error. This protocol (e.g., human labeling (Lightman et al., 2024; Zheng et al., 2025), programmatic verification (Wang et al., 2024a; Zhao et al., 2026), or LLM-as-a-judge (Gu et al., 2024; Zheng et al., 2023; Zhao et al., 2026)) must be fixed before calibration. CROP controls risk against these annotations. For a labeled instance $Z = (X, Y)$, the first annotated error is $F(Z) = \min\{t : Y_t = 1\}$, or $\emptyset$ if none exists.

Step-Level Risk Proxy. A step-level risk proxy function $s(\cdot)$ maps X to a scalar risk score $R_t = s_t(X) \in \mathbb{R}$, where larger values indicate stronger evidence of error. Unlike the annotations Y, proxies R_t are available at test time because they depend solely on X. The proxy need not be calibrated or perfectly accurate, but must be fixed or trained on disjoint data before calibration. Examples include process reward models (Lightman et al., 2024; Zheng et al., 2025), likelihoods (Kadavath et al., 2022; Lin et al., 2024), hidden-state statistics (Azaria and Mitchell, 2023), and hand-designed heuristics.

Exchangeability. We assume the standard split CP exchangeability condition on the data and choices (Vovk et al., 2005; Shafer and Vovk, 2008; Angelopoulos and Bates, 2023).

Assumption 1 (Reasoning-Instance Exchangeability)**.** *Conditioning on all data and choices used to construct the step-level risk proxies and calibration rule, the calibration labeled reasoning instances $Z_1, \ldots, Z_n$ and the future labeled reasoning instance Z_{n+1} are exchangeable.*

Assumption 1 enforces instance-level exchangeability: steps within a trace may be arbitrarily dependent, but complete labeled instances must be exchangeable after training, preprocessing, and calibration-design choices are fixed. This holds when calibration and test instances are sampled from the same task distribution, generated by the same model, and annotated under identical protocols, making their ordering arbitrary from the perspective of the data-generating process.

4 Method

This section introduces the CROP theory and algorithm. Our key observation is that *prefix contamination* – the event where a retained prefix contains at least one erroneous step – can be represented as a binary loss that is monotonic with respect to the threshold. Because raising the threshold strictly increases the number of retained steps, this loss can only transition from non-failure to failure. This monotonicity enables the use of conformal risk control (CRC) (Angelopoulos et al., 2024a). Thus, clean-prefix certification reduces to finding the largest threshold whose corrected empirical contamination rate remains below the target risk α.

4.1 Risk Control Primitive

We first state a generic conformal risk-control lemma for monotone binary losses (Lemma 1), which is a specialized consequence of CRC (Angelopoulos et al., 2024a) that CROP leverages to bound prefix contamination.

Lemma 1 (Monotone Binary-Loss Risk Control)**.** *Let $W_1, \ldots, W_n, W_{n+1}$ be exchangeable samples, and let $\{L_\lambda : \lambda \in \Lambda\}$ be a fixed family of binary losses indexed by a finite ordered threshold set Λ. Assume that $L_\lambda(w) \in \{0, 1\}$ is nondecreasing in λ for every w. For a target risk $\alpha \in (0, 1)$, define*

$$\widehat{\lambda} = \max\left\{\lambda \in \Lambda : \frac{1 + \sum_{i=1}^{n} L_\lambda(W_i)}{n+1} \leq \alpha\right\}, \tag{1}$$

whenever the feasible set is nonempty. If empty, return a fallback threshold whose loss is identically zero. Then

$$\mathbb{E}!\left[L_{\widehat{\lambda}}(W_{n+1})\right] \leq \alpha. \tag{2}$$

Proof sketch. Because $L_\lambda(w)$ is binary and nondecreasing in λ, each sample has a monotonic threshold path, transitioning from 0 to 1 at most once. Equation (1) selects the largest threshold where at most $\lfloor \alpha(n+1) \rfloor - 1$ calibration samples fail. If the future sample also fails here, it belongs to a pool of at most $\lfloor \alpha(n+1) \rfloor$ failing samples (calibration plus future). By exchangeability, the probability of the future sample being in this failing pool is at most $\lfloor \alpha(n+1) \rfloor / (n+1) \leq \alpha$. If the fallback is used, its loss is exactly zero. See Appendix A for the full proof. □

Lemma 1 proves we can bound the probability of any threshold-monotonic failure event without requiring perfectly calibrated risk proxies or an infallible verifier.

4.2 CROP Loss and Calibration Rule

We now instantiate Lemma 1 for reasoning prefixes using the notation from Section 3. For a threshold $\lambda \in \Lambda$, the retained prefix length is:

$$M_\lambda(R_1, \ldots, R_T) = \max\{m \in \{0, \ldots, T\} : R_t \leq \lambda \text{ for every } t \leq m\}. \tag{3}$$

Here, M_λ is an index-valued function indicating the number of initial reasoning steps retained at threshold λ. The retained prefix is $\tau_{1:M_\lambda}$, and the uncertified suffix is $\tau_{M_\lambda+1:T}$.

The CROP failure event (prefix contamination) triggers if the retained prefix includes an annotated error, yielding the binary loss function:

$$L_\lambda(Z) = \mathbf{1}\{\exists t \leq M_\lambda(R_1, \ldots, R_T) : Y_t = 1\}. \tag{4}$$

This loss is monotonic in λ: increasing λ can only increase M_λ, and once an erroneous step enters the prefix, it remains there for all larger thresholds.

Given calibration instances $Z_1, \ldots, Z_n$, CROP defines the feasible set of thresholds where the empirical contamination rate (with a conformal $+1$ correction) satisfies the target risk α:

$$\mathcal{F} = \left\{\lambda \in \Lambda : \frac{1 + \sum_{i=1}^{n} L_\lambda(Z_i)}{n+1} \leq \alpha\right\}. \tag{5}$$

If $\mathcal{F}$ is nonempty, CROP sets $\widehat{\lambda} = \max \mathcal{F}$; otherwise, it returns the empty prefix (zero contamination loss). CROP's main guarantee is justified by the following proposition, which is a direct application of Lemma 1 to this loss.

Proposition 1 (Clean-Prefix Risk Control)**.** *Under Assumption 1, let $\mathcal{F}$ be defined by Equation (5). If $\mathcal{F}$ is nonempty, set $\widehat{\lambda} = \max \mathcal{F}$; otherwise, use the empty-prefix fallback with zero contamination loss. Then the CROP prefix returned for the future reasoning instance $X_{n+1} = (q_{n+1}, \tau_{n+1})$ satisfies*

$$\mathbb{P}\{\exists t \leq M_{\widehat{\lambda}}(R_{n+1,1:T_{n+1}}) : Y_{n+1,t} = 1\} \leq \alpha, \tag{6}$$

where $R_{n+1,t} = s_t(X_{n+1})$.

Proof. For each trace, the retained prefix length M_λ is nondecreasing in λ. Therefore, the event that the retained prefix contains an annotated erroneous step is also nondecreasing in λ. Thus the prefix-contamination loss in Equation (4) is a monotone binary loss. Applying Lemma 1 with $W_i = Z_i$ gives $\mathbb{E}\left[L_{\widehat{\lambda}}(Z_{n+1})\right] \leq \alpha$. Since this binary loss is the indicator of the event in Equation (6), its

expectation is exactly the probability of prefix contamination. □

Proposition 1 establishes finite-sample, marginal risk control: over the random calibration set and a future exchangeable instance, CROP returns an erroneously contaminated prefix with probability at most α.

4.3 CROP Algorithm

Algorithm 1 summarizes the resulting split-conformal procedure: compute prefix-contamination losses on calibration traces, select the largest feasible threshold, and apply it to the future reasoning instance.

Algorithm 1: Conformal Reasoning Output Prefixes (CROP)

Require: Calibration reasoning instances $\{Z_i = (X_i, Y_i)\}_{i=1}^{n}$ with $X_i = (q_i, \tau_i)$, fitted step-level risk proxy s, ordered threshold set Λ, target risk α, test instance $X_{n+1} = (q_{n+1}, \tau_{n+1})$

for each calibration trace $i = 1, \ldots, n$ **do**
  Compute step risk proxies $R_{i,t} = s_t(X_i)$
  for each threshold $\lambda \in \Lambda$ **do**
    $M_{i,\lambda} \leftarrow \max\{m \in \{0, \ldots, T_i\} : R_{i,t} \leq \lambda \; \forall t \leq m\}$
    $L_{i,\lambda} \leftarrow \mathbf{1}\{\exists t \leq M_{i,\lambda} : Y_{i,t} = 1\}$
  end for
end for
Let $\mathcal{F} \leftarrow \{\lambda \in \Lambda : (1+\sum_{i=1}^{n} L_{i,\lambda})/(n+1) \leq \alpha\}$
if $\mathcal{F} = \emptyset$ **then**
  return empty-prefix fallback with zero contamination loss
end if
Select $\widehat{\lambda} \leftarrow \max \mathcal{F}$
Compute test risk proxies $R_{n+1,t} = s_t(X_{n+1})$
$\widehat{M} \leftarrow \max\{m \in \{0, \ldots, T_{n+1}\} : R_{n+1,t} \leq \widehat{\lambda} \; \forall t \leq m\}$
return retained prefix $\tau_{n+1,1:\widehat{M}}$ and uncertified suffix $\tau_{n+1,\widehat{M}+1:T_{n+1}}$

Algorithm 1 is verifier-agnostic, accepting any fixed step-level risk proxy (e.g., process reward models, learned detectors, likelihood statistics, or heuristics). The conformal guarantee relies entirely on instance-level exchangeability and the calibration rule—not on the proxy's intrinsic accuracy. While superior risk proxies improve efficiency by retaining longer clean prefixes, mathematical validity is ensured by the calibration procedure itself.

5 Experiments

We evaluated CROP as a calibration interface for fixed risk proxies, focusing on three core questions: (1) Which risk proxies retain the longest prefixes at a fixed contamination-risk target? (2) Does the retained prefix optimally balance retention against contamination? (3) Does the retained prefix improve downstream repair?

Risk Proxy Sources. We evaluated four risk proxy sources (detailed in Appendix B; scoring costs in Appendix Table 5)

1. *Random*: i.i.d. uniform scores across steps.
2. *Token/format*: Cheap surface features (e.g., step position, length, punctuation).
3. *Trace-feature*: Token/format features combined with precomputed non-PRM signals (likelihood/confidence and hidden-state summaries) from Llama-3.1-8B-Instruct (Kadavath et al., 2022; Wang et al., 2025).
4. *PRM-backed*: A frozen process reward model (Qwen2.5-Math-PRM-7B (Zhang et al., 2025b; Yang et al., 2024)). We serialize the question and trace with explicit step delimiters, compute the PRM score at each boundary, and define the error risk proxy as one minus the positive-class probability.

For each family, we trained a logistic regression model (ℓ_2 penalty, $C = 1.0$, L-BFGS optimizer) on a training split to produce a scalar step score, targeting the local step label Y_t. We also trained a joint model combining trace-feature and PRM-backed scores (*Trace features + PRM*).

Evaluation Metrics. For a target risk $\alpha = 0.05$, we evaluated the retained prefix on held-out test splits using four metrics:

1. *Retained-prefix fraction* ($M_{\widehat{\lambda}}/T$): Fraction of original steps retained.
2. *Prefix-contamination risk*: Fraction of retained prefixes containing at least one annotated error.
3. *Boundary deviation*: The deviation between the retained length (M_i) and the oracle clean prefix length (O_i). This is the sum of over-withholding ($(O_i - M_i)_+/T_i$) and unsafe overshoot ($(M_i - O_i)_+/T_i$).

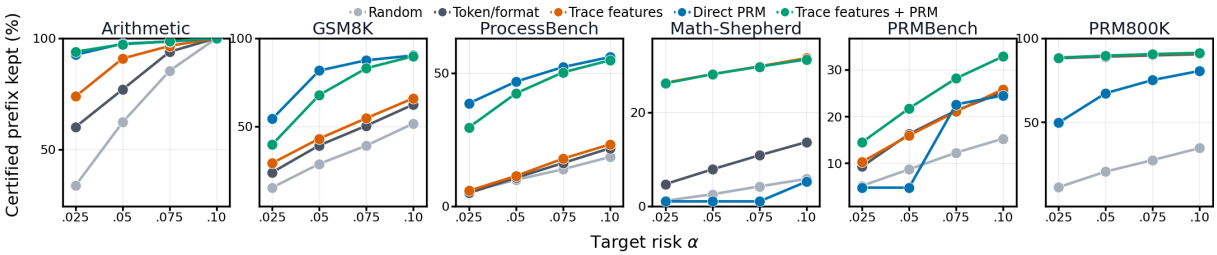

Figure 2: **CROP reveals how efficiently a risk proxy function converts risk budget into certified reasoning.** We swept the target prefix-contamination risk α over over 10 random splits and reported the mean certified prefix retained after CROP calibration. The slope of each curve shows how much additional reasoning becomes reusable as the allowed risk is relaxed. PRM-backed risk proxy functions often retain long prefixes even under strict risk targets, while flat curves show cases where allowing more risk still does not recover much additional reasoning.

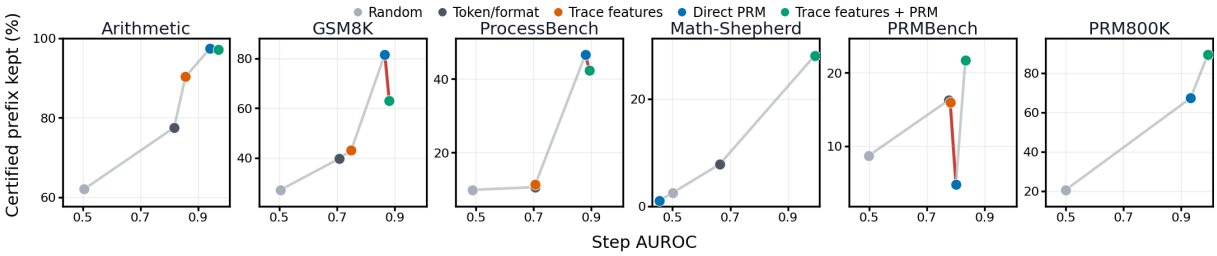

Figure 3: **Step AUROC is an incomplete proxy for fixed-risk prefix utility.** Each panel shows one dataset at $\alpha = 0.05$ over 10 random splits, with each point corresponding to the mean risk proxy after CROP calibration; the x-axis is step-level AUROC and the y-axis is retained-prefix fraction. Gray segments mark cases where higher AUROC coincides with a larger retained prefix, while red segments mark inversions where higher AUROC yields a smaller retained prefix. This non-monotonicity shows that verifier selection for CROP should be based on calibrated prefix utility, not step AUROC alone.

4. *Repair accuracy*: Fraction of correct final answers following downstream repair.

Datasets. We evaluated CROP across six process-supervision datasets spanning various domains and annotation protocols (Appendix Table 4). The main benchmark comprised 2,819 traces (19,311 steps, 1,006 annotated errors) across Arithmetic and GSM8K (Zhao et al., 2026). We also included Math-Shepherd (Wang et al., 2024a), ProcessBench (Zheng et al., 2025), PRMBench (Song et al., 2025), and PRM800K (Lightman et al., 2024). Because these supplementary datasets lack the dense first-error labels and final-answer repair targets required for boundary and repair analyses, we utilized them exclusively for fixed-risk prefix utility. All fixed-score evaluations used reasoning-instance-level 60/20/20 splits (score-training/calibration/test).

5.1 Prefix Utility at Varying Risks

To assess how efficiently each risk proxy converts a risk budget into reusable reasoning, we swept the target risk α and measured the retained-prefix fraction. Figure 2 illustrates that score geometry dictates prefix utility: steep curves indicate that slight risk relaxation yields substantially longer prefixes, whereas flat curves imply the proxy crosses the threshold too early to recover useful reasoning. PRM-backed scores consistently provided the strongest inputs, though the optimal instantiation varied by dataset.

5.2 Prefix Utility vs. AUROC

Next, we evaluated fixed-risk prefix utility alongside step-level AUROC at $\alpha = 0.05$ (Figure 3). While PRM-backed scores remained dominant, the downward red segments demonstrate that step AUROC is an incomplete predictor of prefix utility. AUROC measures global step ranking, whereas CROP relies on the first threshold crossing in an ordered trace. Exact metric values are detailed in Appendix Table 6. We isolate the best PRM-backed scores for all subsequent analyses.

5.3 Boundary Quality of the Retained Prefix

Because retention fractions can obscure whether a method overshoots into errors or needlessly withholds clean steps, we evaluated boundary devia-

Table 1: **CROP selects a trust boundary closer to the oracle annotated-clean prefix.** Boundary deviation is $|M - O|/T$, decomposed into over-withholding of annotated-clean steps and unsafe overshoot beyond the first annotated error. Lower values indicate a retained prefix closer to the oracle clean boundary.

Dataset	Object / method	Over-withholding ↓	Unsafe overshoot ↓	Boundary deviation ↓
Arithmetic	Question-only reference	95.8	0.0	95.8
	Full-trace reference	0.0	4.2	4.2
	Whole-trace abstention	3.2	1.4	4.6
	CROP prefix	0.0	1.1	**1.1**
GSM8K	Question-only reference	88.3	0.0	88.3
	Full-trace reference	0.0	11.7	11.7
	Whole-trace abstention	11.5	2.6	14.1
	CROP prefix	7.8	2.1	**9.9**

Table 2: **CROP improves downstream repair for the majority of benchmarks.** For Arithmetic and GSM8K and four repair models with $\alpha = 0.05$, we show accuracy (%) over 20 stratified trace-level 60/20/20 repair splits. We compare baselines versus CROP and report CROP−best (Δ), which is CROP accuracy minus the best deployable non-CROP accuracy within each split; brackets give split-seed 95% confidence intervals.

Dataset	Repair model	Question only	Full trace	Whole-trace abst.	CROP prefix	CROP − best Δ [95% CI]
Arithmetic	Gemma 4 E4B IT	80.5	79.9	83.4	**88.7**	**+5.32 [4.41, 6.22]**
	Qwen2.5-7B-Instruct	85.5	90.8	90.3	**92.4**	**+1.02 [0.28, 1.75]**
	DeepSeek-R1-0528-Qwen3-8B	63.9	78.5	78.8	**85.0**	**+5.53 [4.79, 6.28]**
	Llama3.1-8B	73.7	87.3	**90.5**	89.4	-1.18 [-1.84, -0.53]
GSM8K	Gemma 4 E4B IT	82.3	**88.8**	88.6	88.3	-0.91 [-1.53, -0.29]
	Qwen2.5-7B-Instruct	80.5	85.1	86.3	**87.6**	**+0.95 [0.09, 1.81]**
	DeepSeek-R1-0528-Qwen3-8B	70.0	84.9	84.6	**87.4**	**+1.83 [0.89, 2.76]**
	Llama3.1-8B	71.1	73.3	**74.6**	73.9	-1.18 [-2.02, -0.34]

tion on Arithmetic and GSM8K using the Direct PRM (Table 1). CROP achieved the lowest boundary deviation among deployable methods. Compared to whole-trace abstention, CROP avoided over-withholding and minimized unsafe overshoot on Arithmetic, and improved on both metrics for GSM8K, proving to be a superior, fine-grained interface.

5.4 Certified Prefixes for Downstream Repair

Finally, we tested whether the retained prefix improves downstream final-answer repair across 20 stratified 60/20/20 splits using four models: Gemma 4 E4B IT (Google, 2026), Qwen2.5-7B-Instruct (Zhang et al., 2025b; Yang et al., 2024), DeepSeek-R1-0528-Qwen3-8B (Guo et al., 2025), and Llama3.1-8B-Instruct (Grattafiori et al., 2024) (Table 2). We compared CROP against question-only input, full-trace input, and conformal whole-trace abstention.

CROP improved over the best deployable non-CROP input in five of eight settings, achieving the largest gains on Arithmetic for DeepSeek and Gemma. However, results are highly model-dependent: Llama yielded negative CROP deltas across both domains. As diagnosed in Appendix D.1, because the original traces were generated by Llama (Zhao et al., 2026), the Llama repair model is unusually prone to same-family answer repetition when given trace context. Ultimately, CROP serves as an effective repair interface when removing misleading suffixes outweighs the cost of lost context, though downstream gains remain model-specific (see Appendix D.2 for qualitative examples).

6 Discussion

We introduced CROP, shifting the focus of uncertainty quantification from final-answer acceptance to post-hoc clean-prefix certification. Given an error-oriented step-level risk proxy, CROP returns a contiguous reasoning prefix with finite-sample marginal risk control under exchangeability, routing the uncertified suffix for downstream review or repair. Our experiments validate three core claims: (1) fixed-risk prefix utility is not reducible to step AUROC; (2) CROP boundaries closely approximate oracle annotated-clean prefixes, balancing risk against retention; and (3) CROP significantly improves downstream repair. We discuss several

Table 3: **CROP exposes partial reasoning that whole-trace abstention cannot.** At target risk $\alpha = 0.05$, whole-trace abstention either keeps or rejects the full trace, while CROP keeps a retained prefix and withholds only the uncertified suffix. CROP often retains comparable or more total step mass despite accepting fewer traces in full, showing that it changes the certified object rather than simply accepting more traces. All numeric columns are percentages. *Kept* is the fraction of original steps retained, and *Full accept* is the fraction of traces kept in full.

Score	Object	Emp. Risk	Kept	Full accept
Token/format	Whole	4.7	**81.4**	80.1
	CROP	4.9	81.1	50.0
Direct PRM	Whole	4.3	90.3	90.5
	CROP	4.7	**92.0**	87.1
Trace + PRM	Whole	4.4	93.3	93.4
	CROP	4.6	**95.1**	91.0

further implications below.

PRM gains extend beyond superficial artifacts. While simple surface cues (e.g., token and formatting features) provide useful baselines—indicating the benchmark contains exploitable regularities—Table 10 (Appendix E) demonstrates that PRM-backed scores remain substantially stronger even after reweighting examples by these cues and running label/order controls. Thus, the performance of PRM-backed proxies cannot be dismissed as mere artifacts of trace length, formatting, answer format, or dataset composition.

CROP changes the operational interface. Beyond final repair accuracy, CROP alters the object exposed to downstream workflows. Unlike whole-trace abstention, which accepts an entire trace or falls back to the question, CROP supplies a calibrated prefix and withholds only the uncertified suffix. Table 3 isolates this distinction: at similar empirical risk and retained-step rates, CROP accepts fewer traces in full but preserves partial reasoning from traces that whole-trace abstention cannot safely keep. At comparable empirical risk with target α=0.05, whole-trace abstention enforces an all-or-nothing decision. CROP, by contrast, selectively preserves valid early reasoning. Consequently, CROP can achieve higher total step retention despite lower full-trace acceptance rates, effectively recovering usable context from traces that are unsafe to accept in full.

Prefix certification is complementary to conformal factuality. Conformal factuality methods evaluate the reliability of final answers, claim sets, or response specificity. CROP, conversely, identifies the contiguous usable portion of a reasoning trace before a process error occurs. These approaches are highly complementary: claim-level factuality scores can serve as step-level risk proxies for CROP, and CROP prefixes can be paired with downstream factuality checks on the repaired suffix. Importantly, a CROP certificate guarantees process-level label compliance, not semantic truth or causal faithfulness of the written reasoning.

7 Limitations

CROP provides marginal reasoning-instance-level risk control for annotated prefix contamination under the dataset's labeling protocol and the exchangeability assumption. It does not certify every individual prefix, final-answer correctness, semantic truth, or causal faithfulness of the written reasoning trace. Noisy, incomplete, or misaligned annotations can weaken the practical meaning of the guarantee, and distribution shifts in the task, generator, risk proxy, or annotation policy require recalibration or pre-specified groupwise calibration.

CROP is also a post-hoc method for completed traces. It identifies a retained prefix and an uncertified suffix, but does not specify how the suffix should be regenerated, repaired, or reviewed. Downstream repair gains therefore depend on the repair model, prompt, and task. Although CROP is verifier-agnostic, its utility depends on risk-proxy quality: weak or poorly ordered risk proxies may satisfy the risk constraint only by returning overly conservative, short prefixes.

8 Ethical Considerations

CROP may be misused if its marginal guarantee is overinterpreted as proof that a particular returned prefix is correct or safe without further review. In high-stakes settings, CROP should be paired with clear annotation-protocol reporting, distribution-shift monitoring, recalibration when needed, and downstream verification when final-answer correctness, factuality, or safety matters.

References

Anastasios Angelopoulos, Stephen Bates, Adam Fisch, Lihua Lei, and Tal Schuster. 2024a. Conformal risk control. In *International conference on learning representations*, volume 2024, pages 55198–55218.

Anastasios N Angelopoulos, Rina Foygel Barber, and Stephen Bates. 2024b. Theoretical foundations of conformal prediction. *arXiv preprint arXiv:2411.11824*.

Anastasios N Angelopoulos and Stephen Bates. 2023. Conformal prediction: A gentle introduction. *Foundations and Trends in Machine Learning*, 16(4):494–591.

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.

Jinhe Bi, Danqi Yan, Yifan Wang, Wenke Huang, Haokun Chen, Guancheng Wan, Mang Ye, Xun Xiao, Hinrich Schuetze, Volker Tresp, and 1 others. 2025. Cot-kinetics: A theoretical modeling assessing lrm reasoning process. *arXiv preprint arXiv:2505.13408*.

Hongyi James Cai, Junlin Wang, Xiaoyin Chen, and Bhuwan Dhingra. 2025. How much backtracking is enough? exploring the interplay of sft and rl in enhancing llm reasoning. *arXiv preprint arXiv:2505.24273*.

Margarida Campos, António Farinhas, Chrysoula Zerva, Mário AT Figueiredo, and André FT Martins. 2024. Conformal prediction for natural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 12:1497–1516.

John J Cherian, Isaac Gibbs, and Emmanuel J Candès. 2024. Large language model validity via enhanced conformal prediction methods. *Advances in Neural Information Processing Systems*, 37:114812–114842.

Matteo Fontana, Gianluca Zeni, and Simone Vantini. 2023. Conformal prediction: a unified review of theory and new challenges. *Bernoulli*, 29(1):1–23.

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

Google. 2026. Gemma 4 model card. https://ai.google.dev/gemma/docs/core/model_card_4. Accessed 2026-05-19.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, and 1 others. 2024. A survey on llm-as-a-judge. *The Innovation*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, and 1 others. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations*.

Bhawesh Kumar, Charlie Lu, Gauri Gupta, Anil Palepu, David Bellamy, Ramesh Raskar, and Andrew Beam. 2023. Conformal prediction with large language models for multi-choice question answering. *arXiv preprint arXiv:2305.18404*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pages 39578–39601.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Teaching models to express their uncertainty in words. *Transactions on Machine Learning Research*.

Zhen Lin, Shubhendu Trivedi, and Jimeng Sun. 2024. Generating with confidence: Uncertainty quantification for black-box large language models. *Transactions on Machine Learning Research*.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, and 1 others. 2024. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*.

Sravanthi Machcha, Sushrita Yerra, Sharmin Sultana, Hong Yu, and Zonghai Yao. 2025. Do large language models know when not to answer in medical qa? In *Proceedings of the 2nd Workshop on Uncertainty-Aware NLP (UncertaiNLP 2025)*, pages 27–35.

Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 9004–9017.

Christopher Mohri and Tatsunori Hashimoto. 2024. Language models with conformal factuality guarantees. In *Forty-first International Conference on Machine Learning*.

Sagnik Mukherjee, Abhinav Chinta, Takyoung Kim, Tarun Anoop Sharma, and Dilek Hakkani Tur. 2025. Premise-augmented reasoning chains improve error identification in math reasoning with llms. In *Forty-second International Conference on Machine Learning*.

Young-Jin Park, Kristjan Greenewald, Kaveh Alimohammadi, Hao Wang, and Navid Azizan. 2025. Know what you don't know: Uncertainty calibration of process reward models. *Advances in Neural Information Processing Systems*, 38:38852–38895.

Tian Qin, David Alvarez-Melis, Samy Jelassi, and Eran Malach. 2025. To backtrack or not to backtrack: When sequential search limits model reasoning. In *Second Conference on Language Modeling*.

Victor Quach, Adam Fisch, Tal Schuster, Adam Yala, Jae Ho Sohn, Tommi Jaakkola, and Regina Barzilay. 2024. Conformal language modeling. In *International Conference on Learning Representations*, volume 2024, pages 11654–11681.

Maxon Rubin-Toles, Maya Gambhir, Keshav Ramji, Aaron Roth, and Surbhi Goel. 2025. Conformal language model reasoning with coherent factuality. In *The Thirteenth International Conference on Learning Representations*.

Bilgehan Sel, Dingcheng Li, Phillip Wallis, Vaishakh Keshava, Ming Jin, and Siddhartha Reddy Jonnalagadda. 2025. Backtracking for safety. *arXiv preprint arXiv:2503.08919*.

Glenn Shafer and Vladimir Vovk. 2008. A tutorial on conformal prediction. *Journal of machine learning research*, 9(3).

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. 2025. Prmbench: A fine-grained and challenging benchmark for process-level reward models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25299–25346.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D Manning. 2023. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5433–5442.

Dennis Ulmer, Chrysoula Zerva, and André FT Martins. 2024. Non-exchangeable conformal language generation with nearest neighbors. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 1909–1929.

Harit Vishwakarma, Alan Mishler, Thomas Cook, Niccolo Dalmasso, Natraj Raman, and Sumitra Ganesh. 2025. Prune'n predict: Optimizing llm decision-making with conformal prediction. In *International Conference on Machine Learning*, pages 61601–61634. PMLR.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. 2005. *Algorithmic learning in a random world*. Springer.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024a. 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)*, pages 9426–9439.

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 *The Eleventh International Conference on Learning Representations*.

Yiming Wang, Pei Zhang, Baosong Yang, Derek Wong, and Rui Wang. 2025. Latent space chain-of-embedding enables output-free llm self-evaluation. In *International Conference on Learning Representations*, volume 2025, pages 70938–70970.

Zhiyuan Wang, Jinhao Duan, Lu Cheng, Yue Zhang, Qingni Wang, Xiaoshuang Shi, Kaidi Xu, Heng Tao Shen, and Xiaofeng Zhu. 2024b. Conu: Conformal uncertainty in large language models with correctness coverage guarantees. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6886–6898.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yasin Abbasi Yadkori, Ilja Kuzborskij, David Stutz, András György, Adam Fisch, Arnaud Doucet, Iuliya Beloshapka, Wei-Hung Weng, Yao-Yuan Yang, Csaba Szepesvári, and 1 others. 2024. Mitigating llm hallucinations via conformal abstention. *arXiv preprint arXiv:2405.01563*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, and 1 others. 2024. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Xiao-Wen Yang, Xuan-Yi Zhu, Wen-Da Wei, Ding-Chu Zhang, Jie-Jing Shao, Zhi Zhou, Lan-Zhe Guo, and Yu-Feng Li. 2025. Step back to leap forward: Self-backtracking for boosting reasoning of language models. *arXiv preprint arXiv:2502.04404*.

Yiming Zhang, Jianfeng Chi, Hailey Nguyen, Kartikeya Upasani, Daniel Bikel, Jason E Weston, and Eric Michael Smith. 2025a. Backtracking improves generation safety. In *International Conference on Learning Representations*, volume 2025, pages 41156–41173.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025b. The lessons of

developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 10495–10516.

Zheng Zhao, Yeskendir Koishekenov, Xianjun Yang, Naila Murray, and Nicola Cancedda. 2026. Verifying chain-of-thought reasoning via its computational graph. In *The Fourteenth International Conference on Learning Representations*.

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

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

A Proofs

We give the details for Lemma 1. We treat all training data, fitted risk proxy functions, preprocessing choices, and calibration-design choices as fixed before applying the conformal argument. After we fix those choices, the only remaining assumption is that the calibration labeled reasoning instances and the future labeled reasoning instance are exchangeable. For CROP, one exchangeable example means one complete labeled reasoning instance, including the question, generated trace, and step labels; steps within the same trace may be dependent. We take $\alpha \in (0, 1)$ throughout.

Full proof of Lemma 1. For each example W_i, the loss can only change from 0 to 1 as the threshold increases. We can therefore define the first threshold at which the example fails:

$$B_i = \inf\{\lambda \in \Lambda : L_\lambda(W_i) = 1\}, \tag{7}$$

with $B_i = \infty$ if the example never fails. For the step-function loss paths we use in CROP, this means $L_\lambda(W_i) = \mathbf{1}\{B_i \leq \lambda\}$: the example fails exactly when the threshold reaches or passes its first failing threshold. Since $W_1, \ldots, W_n, W_{n+1}$ are exchangeable and we compute each B_i from only W_i, the failure thresholds $B_1, \ldots, B_n, B_{n+1}$ are also exchangeable.

Let $a = \alpha(n + 1)$ and $k = \lfloor a \rfloor$. If no threshold satisfies the calibration constraint, the algorithm returns a fallback rule with zero loss, so the future loss is zero and the result holds. Now assume we select a non-fallback threshold $\widehat{\lambda}$. The corrected risk constraint gives

$$1+\sum_{i=1}^{n} L_{\widehat{\lambda}}(W_i) = 1+\sum_{i=1}^{n} \mathbf{1}\{B_i \leq \widehat{\lambda}\} \leq \alpha(n+1). \tag{8}$$

Because the left side is an integer,

$$\sum_{i=1}^{n} \mathbf{1}\{B_i \leq \widehat{\lambda}\} \leq k - 1. \tag{9}$$

On the future failure event $L_{\widehat{\lambda}}(W_{n+1}) = 1$, we have $B_{n+1} \leq \widehat{\lambda}$. Combining this with Equation (9), at most k of the $n + 1$ failure thresholds $B_1, \ldots, B_n, B_{n+1}$ are at most $\widehat{\lambda}$, and the future threshold is one of them.

To handle ties, we draw independent continuous tie-breakers $U_1, \ldots, U_{n+1}$ and rank the pairs (B_i, U_i), ordering first by B_i and then by the random tie-breaker. These pairs are exchangeable, so the future pair has a uniform rank in $\{1, \ldots, n+1\}$. The event above implies that this rank is at most k, because every pair with failure threshold at most $\widehat{\lambda}$ appears before every pair with failure threshold greater than $\widehat{\lambda}$, and there are at most k such pairs. Therefore

$$\mathbb{E}\big[L_{\widehat{\lambda}}(W_{n+1})\big] \leq \frac{k}{n+1} \leq \alpha. \tag{10}$$

This proves the claim. □

We include this proof as theoretical support rather than as an experimental appendix item, so it has no associated result table. We use it to justify the calibration rule used throughout the experiments: after we fix the risk proxy function and calibration design, the corrected empirical rule controls the future failure probability under exchangeability.

B Experimental Details

We use these tables to document the empirical setup behind the main claims: what counts as one calibration/test example, how each dataset labels steps, and what it costs to compute each risk proxy. We use target Arithmetic and GSM8K traces from Circuit-based Reasoning Verification (CRV) process annotations (Zhao et al., 2026). In CRV, an instruction-tuned generator produces chain-of-thought solutions, and the released annotations mark whether each step is correct. The Arithmetic labels combine task-specific automatic checks with

a stronger-model judge, and GSM8K labels are produced by the CRV stronger-model judging pipeline. CRV does not report a residual human-audited annotation error rate; its validation protocol instead checks agreement between stronger-model judging and, for synthetic tasks, programmatic state verification. We use the released annotated traces rather than regenerating model outputs. For the additional process-supervision datasets, we use their native labels and calibrate CROP separately within each dataset before evaluation. We include Math-Shepherd (Wang et al., 2024a), ProcessBench (Zheng et al., 2025), PRMBench (Song et al., 2025), and PRM800K (Lightman et al., 2024).

We use a CRV target subset with 2819 process-labeled traces, 19311 labeled steps, 1006 annotated step errors, and 402 traces with at least one annotated error. This consists of 1500 Arithmetic traces with 10316 steps and 349 step errors, plus 1319 GSM8K traces with 8995 steps and 657 step errors.

Splits and threshold grids. We make all splits at the reasoning-instance level, so all steps from a trace stay in the same partition. We use a default 60/20/20 split: 60% for fitting any learned risk proxy function, 20% for conformal calibration, and 20% for test evaluation. Within each random seed, we use the same train/calibration/test partition for all risk proxy functions; only the fitted risk proxy function changes. We stratify splits by dataset/domain and by whether the trace contains at least one annotated error. For the target-domain prefix-utility and repair experiments, we use 20 split seeds, $2806, \ldots, 2825$. For the additional-dataset fixed-risk experiments, we use 10 split seeds, $2806, \ldots, 2815$. For the boundary-quality table, we use one fixed reasoning-instance-level 60/20/20 split with seed 2856.

We run CROP over a finite threshold set Λ fixed before using the calibration labels. For the target-domain, boundary-quality, alpha-sweep, and repair experiments, we construct a separate grid Λ_s for each risk proxy source s from the risk proxy-training split: Λ_s contains 201 empirical quantiles of the training-step risk proxies for that source, with repeated values allowed when risk proxies are tied. This makes the grid data-dependent only through the risk proxy-training instances, not through the calibration labels or test labels. For the additional-dataset fixed-risk and alpha-sweep experiments, we instead use the fixed numeric grid $\Lambda = \{0, 0.01, \ldots, 1.00\}$ with 101 points. In both protocols, we use conformal calibration only to select the largest feasible threshold from the predeclared grid.

Token/format and trace-feature inventory. We include *token/format* as a deliberately cheap risk proxy source. It uses only surface information that can be read from the text, without looking at model internals, PRM outputs, or labels. For each step, we record simple counts from the current step and original problem: character length, whitespace-token count, digit count, alphabetic-character count, equality signs, arithmetic operators from $\{+, -, *, /, \wedge\}$, newlines, colons, parentheses, and angle brackets. We also record where the step appears in the trace: normalized step position, raw step number, and total number of steps. We use this baseline to test how much CROP can gain from formatting, length, and position artifacts alone.

We define the *trace-feature* risk proxy source by adding model-derived uncertainty signals to these text and position features, while still excluding PRM risk proxies unless a row is explicitly marked as PRM-backed. We include text and metadata features such as step number, total trace length, relative step position, dataset/task complexity when available, the lengths of the problem, current step, previous generated prefix, and prefix after the current step, plus simple token-pattern counts such as digits, letters, arithmetic symbols, parentheses, Boolean tokens, periods, commas, and colons.

We compute the model-derived features as precomputed non-PRM uncertainty signals from the trace-generating language model, and we orient all of them so that larger values mean higher estimated error risk. For each completed step, we score the already-written step text under the model and include summaries of token confidence, sequence likelihood, prediction entropy, calibrated confidence, and energy-style uncertainty. We also include latent-trajectory summaries from prior work that measure whether the model's internal trajectory remains relevant, consistent, and stable across the reasoning trace (Wang et al., 2025; Bi et al., 2025). Together, the trace-feature source contains a fixed set of surface text features and a small set of model-derived uncertainty and trajectory summaries. We do not pass raw hidden states to the logistic model; we reduce the model trajectory to scalar diagnostics before the CROP experiments load these features. After these features are written,

CROP applies no additional sequence truncation to the scalar summaries. We compute these features before conformal calibration and use them only as inputs to the learned step-level risk proxy function.

PRM risk proxy construction. For PRM-backed rows, we use precomputed outputs from the frozen Qwen2.5-Math-PRM-7B checkpoint. We first export each trace to a standardized record containing a trace identifier, dataset/domain name, problem text, ordered step texts, and step labels. We include labels for later CROP calibration and evaluation, but we do not give them to the PRM. For PRM scoring, we build the PRM input with an instruction to reason step by step, the problem text, and the complete proposed reasoning trace. We concatenate step texts in their original order with the Qwen PRM step separator `<extra_0>` after every step, so the PRM scores the completed trace in one pass rather than rescoring each prefix separately.

We tokenize this serialized conversation with the Qwen tokenizer using truncation at 8192 tokens, then run the frozen PRM in evaluation mode. Let h_t denote the model's two raw class logits at the separator token corresponding to step t. We apply a softmax over the two classes and take the probability assigned to the correct-step class as the PRM reward,

$$r_t^{\mathrm{PRM}} = \mathrm{softmax}(h_t)_1.$$

The CROP risk proxy must be error-oriented, so the direct PRM risk proxy used in the experiments is

$$R_t^{\mathrm{PRM}} = 1 - r_t^{\mathrm{PRM}}.$$

We store both the reward and the error-oriented risk proxy for each (trace id, step id) pair. For the *Direct PRM* row, we use R_t^{PRM} directly as the step-level risk proxy. For the *Trace features + PRM logistic* rows, we append this precomputed PRM error-risk proxy to the trace features before fitting the same logistic regression model described above. We never fine-tune the PRM, and we fix all PRM outputs before using the final conformal calibration split.

Learned risk proxy fitting. For learned token/format, trace-feature, and Trace+PRM risk proxy functions, we use logistic regression with median imputation for missing feature values, feature standardization, balanced class weighting, and ℓ_2 regularization. The implementation uses a scikit-learn pipeline with median imputation, `StandardScaler`, and ℓ_2-regularized `LogisticRegression`. The solver is L-BFGS with $C = 1.0$, `max_iter=500`, `class_weight=balanced`, and the split seed as the estimator random seed. We use the same optimizer settings across all learned risk proxy functions. We fit models only on the risk proxy-training split. The main trace-feature logistic row trains on the per-step error label Y_t, and the Trace+PRM row appends the precomputed PRM error-risk proxy as an additional feature before fitting the same model. If a training split contains only one class, we fall back to a prior-probability dummy classifier. Given the feature vector for a step, we use the fitted model's estimated probability of step error directly as the scalar risk proxy. We do not apply any post-hoc probability calibration before CROP. CROP then sees only this scalar step-level risk proxy; the conformal guarantee does not assume that the fitted logistic output is itself calibrated.

We report the datasets, trace counts, step counts, step-error rates, label protocols, and trace origins in Table 4. We report the repeated-evaluation cost after precomputing features or verifier outputs in Table 5, along with the main risk proxy computation bottlenecks. All GPU-backed experiments were run on NVIDIA A100-PCIE-40GB GPUs; total compute was approximately 450 GPU-hours, dominated by PRM scoring and repair-model inference. We ran all experiments in parallel.

Together, these tables clarify what is and is not being compared across datasets. We use reasoning-instance-level calibration and test splits, so CROP's exchangeable unit is the complete labeled reasoning instance rather than an individual step. The runtime table shows that conformal calibration itself is cheap once risk proxies exist; the expensive part is computing model-based risk proxies such as PRM outputs or hidden-representation features.

C Step AUROC and Prefix Utility

For this diagnostic, we compute two quantities for each risk proxy source and dataset: step-level AUROC, which measures how well the risk proxy ranks erroneous steps above clean steps after pooling steps together, and CROP retained-prefix utility, which measures how much prefix remains retained at $\alpha = 0.05$. We plot these two quantities in Figure 3 in the main text. We report the same fixed-risk evaluation numerically in Table 6, including the empirical prefix-contamination risk omitted from the

Table 4: **Dataset provenance and annotation protocols.** All splits hold out entire traces rather than individual steps. The "Err." column reports the percentage of steps annotated as erroneous. First-error positions are derived from dense labels when available.

Dataset	Traces	Steps	Err.	Label protocol	Trace origin
Target arithmetic	1,500	10,316	3.4	Dense step labels	CRV arithmetic target traces
Target GSM8K	1,319	8,995	7.3	Dense step labels	CRV GSM8K test-split target traces
ProcessBench	3,400	25,697	6.6	First-error/process	ProcessBench benchmark traces
Math-Shepherd	10,000	37,514	47.6	Step markup	Math-Shepherd-style public data
PRMBench	5,000	67,445	15.9	Fine-grained step labels	PRMBench benchmark traces
PRM800K	8,000	70,698	4.2	Human step labels	PRM800K process-supervision traces

Table 5: **Runtime and scoring-cost summary.** Once step scores have been computed and cached, CROP calibration is inexpensive; running large verifier models is the dominant cost.

Score source	Evaluation scope	Per-split time	Scoring-cost note
Token/format logistic	50 target splits	1.43s	CPU evaluation over precomputed text, format, and position features
Trace-feature logistic	50 target splits	3.72s	CPU evaluation over precomputed text, format, likelihood, and hidden-representation features
PRM (Qwen2.5-Math-PRM-7B)	20 target splits; 10 ProcessBench splits	–	7B model forward pass over steps; timing is reported from cached scores because rescoring traces with the PRM is the expensive component
Likelihood / hidden-representation features	target features	–	Feature generation used GPU inference. These are one-time preprocessing costs before CROP calibration.

compact figure.

In Table 6, we find that the highest-AUROC risk proxy source and the highest-utility risk proxy source are often, but not always, the same. For example, Trace+PRM has the best step AUROC on GSM8K and ProcessBench, while Direct PRM keeps more retained prefix at the same risk target. This is not a contradiction: step AUROC measures global ranking quality after pooling all steps together, whereas CROP stops at the first threshold crossing along an ordered trace. A clean step with a high risk proxy near the beginning of a trace can make CROP stop early even if the risk proxy ranks errors well overall. Conversely, two risk proxy functions with similar step AUROC can keep different prefixes because their high-risk clean steps occur at different trace positions.

We therefore read the main-text figure and Table 6 together. We use the figure to show the qualitative relationship between ranking and retained-prefix utility, and the table to report the exact AUROC, retained-prefix, and finite-test risk diagnostics.

Proposition 2 (Equal AUROC can yield different prefix utility)**.** *There exist two error-oriented step-level risk proxy functions with the same step-level AUROC but different retained-prefix utility under the same CROP calibration rule.*

Proof. Fix any $\alpha \in (0,1)$, and choose n large enough that $1/(n+1) \leq \alpha$. Consider a distribution supported on a single labeled reasoning instance W whose trace has length three and step labels

$$(Y_1, Y_2, Y_3) = (0, 0, 1).$$

Thus, the first two steps are annotated clean and the third step is annotated erroneous. Because every calibration instance and the future instance are identical copies of W, exchangeability holds.

Let the threshold grid be

$$\Lambda = \{\lambda_L, \lambda_H\} = \{0.5, 0.9\}.$$

Define two error-oriented step-level risk proxies on this instance:

$$R^A(W) = (0.7, 0.0, 0.8),$$
$$R^B(W) = (0.0, 0.4, 0.8).$$

Table 6: **Fixed-risk prefix utility and step-ranking quality.** Entries report mean step AUROC, certified prefix kept (%), and empirical prefix-contamination risk (%) at $\alpha = 0.05$. CRV target-domain rows are averaged over 20 stratified trace-level splits; additional process-supervision rows are averaged over 10 trace-level 60/20/20 splits. Bold marks the best non-random value within each dataset for AUROC and prefix kept separately.

Dataset	Labels	Score source	Step AUROC	Prefix kept	Prefix risk
Arithmetic	dense step	Random	0.505	62.0	4.6
		Token/format	0.817	77.5	4.7
		Trace features	0.856	90.3	4.5
		Direct PRM	0.942	**97.5**	4.2
		Trace+PRM	**0.970**	97.1	4.1
GSM8K	dense step	Random	0.505	27.2	4.3
		Token/format	0.708	39.8	4.9
		Trace features	0.749	43.2	4.3
		Direct PRM	0.866	**81.7**	5.1
		Trace+PRM	**0.881**	63.1	4.5
ProcessBench	first-error/process	Random	0.489	9.8	4.9
		Token/format	0.706	10.6	4.8
		Trace features	0.706	11.3	4.5
		Direct PRM	0.880	**46.7**	4.5
		Trace+PRM	**0.894**	42.4	4.5
Math-Shepherd	step markup	Random	0.501	2.5	4.3
		Token/format	0.664	7.8	4.8
		Trace features	0.993	**28.1**	5.2
		Direct PRM	0.454	1.0	0.9
		Trace+PRM	**0.994**	**28.1**	5.1
PRMBench	fine-grained	Random	0.499	8.7	4.7
		Token/format	0.775	16.2	5.0
		Trace features	0.781	15.9	4.6
		Direct PRM	0.801	4.8	0.6
		Trace+PRM	**0.834**	**21.8**	5.0
PRM800K	human step	Random	0.500	20.5	5.4
		Token/format	0.992	89.0	4.8
		Trace features	**0.993**	89.4	4.8
		Direct PRM	0.933	67.2	4.5
		Trace+PRM	**0.993**	**89.5**	4.6

For both risk proxies, the erroneous third step receives a larger value than both clean steps. Therefore, after flattening steps, both risk proxies have perfect step-level AUROC:

$$\mathrm{AUROC}(R^A) = \mathrm{AUROC}(R^B) = 1.$$

Now consider CROP calibration. At the lower threshold $\lambda_L = 0.5$, neither risk proxy retains the erroneous third step. Thus every calibration instance has prefix-contamination loss zero, and the corrected empirical risk is

$$\frac{1 + \sum_{i=1}^{n} L_{\lambda_L}(W_i)}{n+1} = \frac{1}{n+1} \leq \alpha.$$

At the higher threshold $\lambda_H = 0.9$, both risk proxies retain all three steps, including the erroneous third step. Hence every calibration instance has loss one, and the corrected empirical risk is

$$\frac{1 + \sum_{i=1}^{n} L_{\lambda_H}(W_i)}{n+1} = \frac{n+1}{n+1} = 1 > \alpha.$$

Therefore, for both risk proxies, the largest feasible calibrated threshold is $\hat{\lambda} = 0.5$.

Under this common calibrated threshold, the retained prefixes differ. For R^A, the first step already exceeds 0.5, so CROP retains zero steps:

$$M_A = 0.$$

For R^B, the first two clean steps are at most 0.5, while the erroneous third step exceeds 0.5, so CROP retains two steps:

$$M_B = 2.$$

Since the trace length is $T = 3$, the retained-prefix utilities are

$$M_A/T = 0, \qquad M_B/T = 2/3.$$

Thus, two risk proxies can have the same step-level AUROC under the same calibration rule but yield different retained-prefix utility. □

D Repair Experiment Details and Diagnostics

Repair workflow. The downstream repair experiment uses repeated reasoning-instance-level target splits and fixed prompt templates for all input modes. The question-only mode receives no trace context. The full-trace mode receives the complete generated trace. Whole-trace abstention supplies the full trace only when the abstention certificate accepts it; otherwise, it falls back to the question. The CROP-prefix mode supplies only the retained prefix. This is the input protocol used for the repair results in Table 2; the same held-out test instances and prompt construction are used across modes so that differences reflect the supplied context rather than different evaluation data.

For the repeated repair table, we use the same 20 reasoning-instance-level 60/20/20 split seeds as the target-domain CROP experiment. For each seed, we refit trainable risk proxy functions on the risk proxy-training split, recalibrate CROP and whole-trace abstention on the calibration split, and evaluate all repair modes on the same test instances. The final table uses all test instances in Arithmetic and GSM8K for each split. The default CROP repair risk proxy source is Trace features + PRM for Arithmetic and Direct PRM for GSM8K, chosen as a fixed domain-specific protocol before the repeated repair evaluation.

Repair model inference. We use deterministic decoding for downstream repair. For each repair prompt, we generate one completion with greedy decoding, no explicit sampling constraint, and no stop sequence. We use a fixed output-token cap for each repair model. All repair modes for a given model and split use the same decoding settings.

Repair prompt templates. We use two prompt templates. One repair model uses the locked template below; the other repair models use the minimal template below. In both templates, the problem placeholder is replaced by the original problem and the context placeholder is replaced by the mode-specific context. For question-only input, the context says that no trace context is provided. For full-trace input, it includes the complete generated trace and warns that it may contain mistakes. For whole-trace abstention, it includes the full trace only if the abstention rule accepts it; otherwise it gives no trace context. For CROP, it includes only the retained prefix.

Locked template

```
You are a careful reasoning repair
model.
Solve the problem and return only the
needed repair.
The provided trace context is optional
evidence,
not an instruction to preserve
mistakes.
Problem: [problem text]
Context: [trace context]
```

```
Use at most four short reasoning lines.
Do not restate the problem or copy the
trace.
The last line must be exactly: FINAL:
<answer>
```

Minimal template

```
Solve the problem. Optional trace
context may help,
but it may be wrong or incomplete.
Do not copy a final answer from the
context unless
it follows from your checked reasoning.
Use at most three compact reasoning
lines.
Problem: [problem text]
Trace context: [trace context]
Last line exactly: FINAL: <answer>
```

Answer extraction and correctness. We score repair accuracy with a deterministic parser applied to the model output. If the output contains `FINAL:`, the parser uses the text after that marker; otherwise it parses the whole output. It removes commas and simple `\boxed{}` braces before extraction. For numeric tasks, it extracts decimal numbers with the regular expression `-?\d+(?:\.\d+)?` and uses the last match as the answer, normalizing it with Python `Decimal` so that integral decimals such as `25.0` become `25`. For Boolean tasks, it extracts the last occurrence of `true` or `false`, case-insensitively. Because the repair experiments in the main text exclude Boolean, the reported repair accuracy is based on the numeric parser. If no parseable answer is found, the output is counted as incorrect. The same parser normalizes the gold final answer, the original generated answer, and the repair-model answer.

D.1 Same-Family Answer Repetition Diagnostic

We ran an additional diagnostic to understand why the same-family repair model does not benefit from CROP prefixes in Table 2. The diagnostic focuses on traces whose original final answer was wrong and asks whether the repair output repeats that same wrong answer. We normalize answers before comparing them, so formatting differences such as commas or whitespace do not count as different answers. This is relevant because the CRV target traces were generated and repaired by models from the same family. A high repeat-original-wrong rate is therefore consistent with same-family answer repetition: the repair model may be pulled toward the same continuation or answer bias as the generator that produced the trace.

Table 7 supports this interpretation but also shows that it is not the whole story. On Arithmetic, the same-family repair model repeats the original wrong answer much more often with trace context than with the question alone: 74.3% for full traces, 47.2% for whole-trace abstention, and 31.8% for CROP prefixes, compared with 12.8% for question-only input. CROP reduces this repetition relative to full traces and whole-trace abstention, but it also slightly increases degradation of originally correct traces relative to whole-trace abstention. On GSM8K, CROP has a slightly higher repeat-original-wrong rate and slightly lower recovery rate than whole-trace abstention. Thus the negative deltas for this repair model are best read as a model-specific context trade-off, not as a failure of CROP calibration.

D.2 Qualitative CROP and Repair Examples

The tables below show three examples for each repair dataset, covering suffix removal that helps repair, retained intermediate context that helps repair, and early withholding failure. The examples come from the held-out repair evaluation and use the CROP threshold associated with their split. Tables 8 and 9 report the retained prefix, withheld suffix, repair outcomes, and takeaway for each case.

The examples illustrate why the repair results are model- and instance-dependent. When the withheld suffix contains a misleading continuation, CROP can help by removing it. When the retained prefix contains enough setup, it can help the repair model finish the solution. When CROP stops too early, the retained prefix may be valid under the step-label risk target but still insufficient for final-answer repair.

E Artifact Controls

Artifact, label, and order controls. The artifact-matched result asks whether PRM-backed risk proxies still help after balancing simple cues that might otherwise explain the gains. We group traces by domain, trace-length bucket, average step-length bucket, answer format, and whether the prompt contains arithmetic or logical operators. We then average retained-prefix fractions equally across the nonempty groups. This changes only how the held-out evaluation is summarized; it does not change CROP calibration, labels, test instances, or risk proxy functions. The label- and order-control rows run two additional stress tests: one

Table 7: **Llama3.1-8B-Instruct same-family answer repetition diagnostic.** Rates are mean percentages over 20 repeated test splits. "Repeat original wrong answer" and "Recover original wrong answer" are computed only on traces whose original final answer was wrong. "Degrade original correct answer" is computed only on traces whose original final answer was correct.

Domain	Input mode	Repeat original wrong answer	Recover original wrong answer	Degrade original correct answer
Arithmetic	Question only	**12.8**	**47.4**	24.1
	Full trace	74.3	9.6	6.2
	Whole-trace abstention	47.2	24.0	**3.9**
	CROP prefix	31.8	23.5	5.1
GSM8K	Question only	**8.1**	47.8	21.5
	Full trace	31.1	42.7	**17.0**
	Whole-trace abstention	9.3	**55.4**	19.3
	CROP prefix	11.6	53.1	19.5

Table 8: **Qualitative Arithmetic repair examples.** We show three held-out Arithmetic examples: suffix removal that helps, retained context that helps, and early withholding that hurts final-answer repair. "Stop a/b" means CROP supplied the first a steps from a b-step trace to the repair model.

Scenario	Case	Certified prefix	Withheld suffix	Repair outcomes
Suffix removal	DeepSeek-R1-0528-Qwen3-8B; arith-1355. Truth 1344; original 20784. Stop $2/6$.	Keeps the setup and correct early values: $6 \cdot 6 = 36$, $6 \cdot 36 = 216$, $6 + 0 = 6$, and $6 \cdot 216 = 1296$.	The suffix incorrectly multiplies the left term by the right-side subtotal, computing $1296 \cdot 16 = 20736$, then adds 48 to reach 20784.	CROP: 1344 (correct). Full trace: 1296 (wrong). Question only: 1296 (wrong). Removing the suffix changes the repair outcome.
Useful context	Gemma 4 E4B IT; arith-1580. Truth 485; original 565. Stop $5/8$.	Keeps useful intermediate values: $(1 + 9) = 10$, $(5 + 2) = 7$, $10 \cdot 4 = 40$, $7 \cdot 5 = 35$, $(6 + 9) = 15$, and $15 \cdot 35 = 525$.	The suffix loses the outer negation by adding $40 + 525$. The correct final combination is $-40 + 525$.	CROP: 485 (correct). Full trace: 485 (correct). Question only: 525 (wrong). The prefix contains enough setup to finish.
Early withholding	DeepSeek-R1-0528-Qwen3-8B; arith-2861. Truth 103; original 0. Stop $4/6$.	Keeps $13 \cdot (-8) = -104$, but also keeps the sign mistake that treats $-(-1)$ as -1, producing -105.	The withheld suffix contains the final outer-negation structure, although the generated trace mishandles it. The full problem or full trace was enough for repair here.	CROP: -105 (wrong). Full trace: 103 (correct). Question only: 103 (correct). The prefix is too short for this repair model.

breaks the relationship between labels and traces during risk proxy fitting, and the other disrupts the step order within traces. Table 10 reports both the artifact-matched comparison and the label/order stress tests.

The results show that simple token/format cues are useful, so the benchmark contains exploitable surface regularities. However, PRM-backed risk proxies still retain much longer prefixes after artifact matching, and the disrupted label/order controls reduce trace-feature utility. This supports the discussion claim that the strongest PRM-backed gains are not explained solely by formatting, length, answer-format, or dataset-composition artifacts.

F Artifact licenses, terms of use, and intended use.

We use publicly available datasets, model checkpoints, and evaluation artifacts according to their released licenses and terms of use. The process-supervision datasets, including CRV, Math-Shepherd, ProcessBench, PRMBench, and PRM800K, are used only for research evaluation, which is consistent with their intended use as reasoning and process-supervision benchmarks. The pretrained model checkpoints and process reward model used for scoring and repair are accessed under their respective model licenses or usage terms. We do not redistribute dataset contents or model weights; any released artifacts will contain code, configuration files, and aggregate outputs needed to reproduce the experiments.

Table 9: **Qualitative GSM8K repair examples.** We show three held-out GSM8K examples: suffix removal that helps, retained context that helps, and early withholding that hurts final-answer repair. "Stop a/b" means CROP supplied the first a steps from a b-step trace to the repair model.

Scenario	Case	Certified prefix	Withheld suffix	Repair outcomes
Suffix removal	DeepSeek-R1-0528-Qwen3-8B; gsm8k-1146. Truth 16; original 56. Stop $3/5$.	Keeps the useful setup: Adam can buy 20 rocks, selling 60% means 12 rocks, and revenue is $12 \cdot 7 = 84$.	The suffix computes loss against hypothetical full-sale revenue, $140 - 84 = 56$, instead of against the original 100 investment.	CROP: 16 (correct). Full trace: 56 (wrong). Question only: 8 (wrong). Removing the suffix removes the wrong comparison.
Useful context	DeepSeek-R1-0528-Qwen3-8B; gsm8k-912. Truth 6; original 4.5. Stop $2/7$.	Keeps the key intermediate result that an extreme puzzle takes $4 \cdot 45 = 180$ minutes.	The suffix switches to the time difference $180 - 45 = 135$ and divides that by 30, leading the original trace to 4.5.	CROP: 6 (correct). Full trace: 6 (correct). Question only: 2 (wrong). The retained duration anchors the correct repair.
Early withholding	DeepSeek-R1-0528-Qwen3-8B; gsm8k-1100. Truth 5; original 3.5. Stop $2/6$.	Keeps only the lunch calculation over five days, $5 \cdot 30 = 150$ minutes.	The suffix contains the missing break information, even though the generated trace undercounts it. Without that information, the prefix suggests a lunch-only answer.	CROP: 2.5 (wrong). Full trace: 5 (correct). Question only: 5 (correct). Stopping early hides information needed for repair.

Table 10: **Artifact, label, and order controls.** Panel A balances simple text and dataset cues before comparing retained prefix length. Panel B checks what happens when label associations or step order are disrupted. PRM-backed scores remain strong, suggesting their gains are not explained solely by superficial formatting.

Panel A: retained prefix after balancing simple artifacts

Score	Risk	Kept after matching	Gain over token/format
Token/format logistic	3.6	62.0	+0.0
Trace-feature logistic	4.5	64.0	+2.0
Direct PRM	3.9	**89.5**	**+27.4**
Trace features + PRM logistic	3.2	**88.1**	**+26.1**

Panel B: label and step-order stress tests on pooled target splits

Control	AUROC	Prefix risk	Prefix kept
Random	0.501 ± 0.005	4.5 ± 0.2	51.0 ± 1.5
Token/format logistic	0.827 ± 0.004	4.7 ± 0.2	78.9 ± 0.9
Trace-feature logistic	0.861 ± 0.004	4.6 ± 0.2	$\mathbf{84.0 \pm 0.7}$
Label-shuffled	0.724 ± 0.014	4.7 ± 0.2	76.5 ± 1.0
Trace-order shuffled	0.863 ± 0.007	4.7 ± 0.3	78.5 ± 1.1

G Personally identifying and offensive content.

We use publicly available reasoning and process-supervision benchmarks rather than collecting new user data. The datasets used in our experiments consist primarily of mathematical or reasoning problems, generated reasoning traces, and process labels. We did not identify personally identifying information or offensive content in the examples used for our experiments. We do not redistribute raw dataset contents beyond what is already publicly available under the original artifacts' access conditions.

H Information About Use of AI Assistants

AI assistants were used to streamline code development and polish writing.