

Conformal Certification of Reasoning Trace Prefixes

Matt Y. Cheung, Ashok Veeraraghavan, Hanjie Chen and Guha Balakrishnan
Rice University, Houston TX, USA

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 and improves 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. Code available at <https://github.com/matthewyccheung/crop>.

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 answer-level UQ compresses this structure into

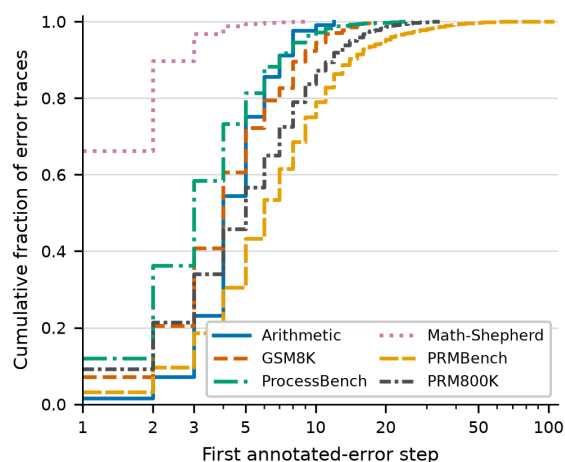


Figure 1: **Erroneous reasoning traces often contain correct and reusable initial steps.** For six diverse reasoning datasets, we plot the empirical CDF of the step index of the first annotated error over all erroneous traces. First errors frequently occur after multiple correct steps, motivating certification and reuse of initial steps rather than treating the entire trace as unusable.

a scalar confidence or binary accept/reject decision (Kadavath et al., 2022; Kuhn et al., 2023; Yadkori et al., 2024), and does not directly identify 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 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 directly address process-level reasoning: certifying an entire trace can discard valid early work

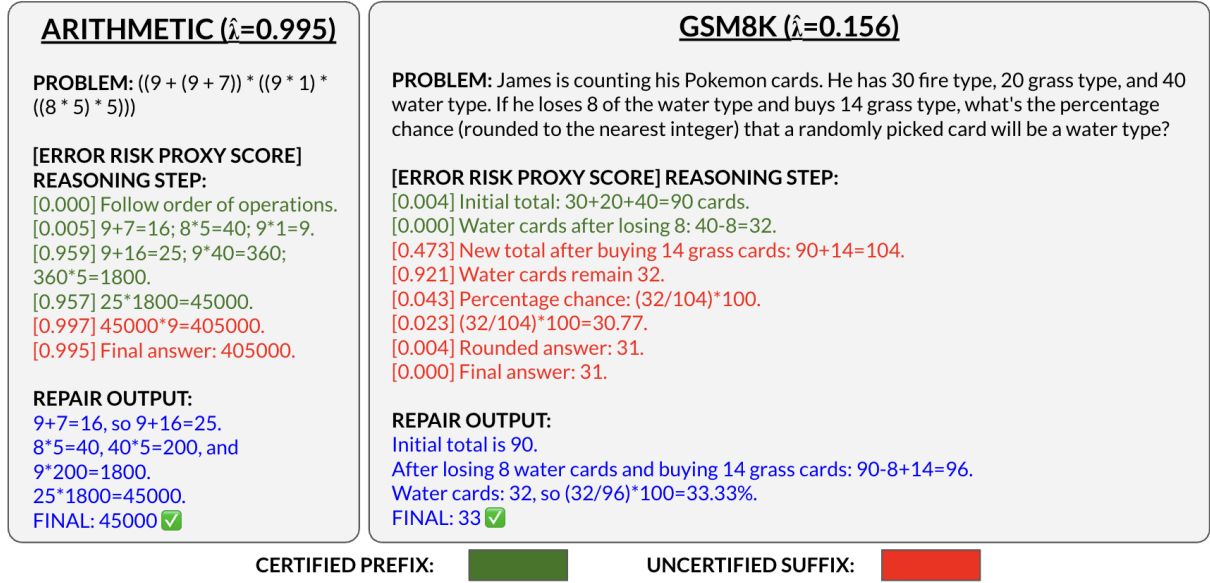


Figure 2: **CROP returns a calibrated prefix of a completed reasoning trace.** We show 2 example traces solving problems from the Arithmetic and GSM8K datasets (Zhao et al., 2026). For each step, CROP computes an error risk proxy score (in square brackets), with larger values indicating greater estimated error risk. Using held-out calibration reasoning instances, CROP selects a domain-specific, conformally calibrated score threshold $\hat{\lambda}$ that controls the marginal probability that the retained prefix contains an annotated error under exchangeability. At test time, CROP returns the longest initial prefix (green) in which no proxy score exceeds the threshold. The remaining suffix (red) is left uncertified and can be routed to downstream review or repair, whose output is shown in blue.

because of a single late mistake, while certifying claims independently does not identify the longest initial sequence of reasoning that can be retained while controlling annotated-error contamination. We refer to this objective as *clean-prefix certification*: selecting a contiguous prefix from the beginning of a reasoning trace while controlling the marginal probability that it contains an annotated 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 2). Given a reasoning instance $X = (q, \tau)$ with question q and trace $\tau = (\tau_1, \dots, \tau_T)$, CROP uses a fixed step-level risk proxy s to assign each step τ_t an error-oriented score $R_t = s_t(X)$. The proxy can be any signal, such as a process reward model, learned detector, or likelihood statistic, with larger scores indicating 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 proxy scores 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 techniques (see Section 2).

We evaluated CROP across six process-labeled reasoning datasets: Arithmetic (Zhao et al., 2026), GSM8K (Zhao et al., 2026), Process-Bench (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 AU-ROC 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 unsafe overshoot (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 strikes a better balance between over- and unsafe overshoot, preserving valid intermediate reasoning and enhancing downstream repair accuracy over uncalibrated UQ baselines on six well-established and diverse benchmarks.

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; Kladny et al., 2025), and whole-trace abstention (Machcha et al., 2025; Yadkori et al., 2024). Sequential methods (Quach et al., 2024; Kladny et al., 2025) construct or filter sets of candidate generations rather than steps within a single reasoning trace. 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 $\tau = (\tau_1, \dots, \tau_T)$ 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 $\tau_{1:m} = (\tau_1, \dots, \tau_m)$, leaving the uncertified suffix $\tau_{m+1:T} = (\tau_{m+1}, \dots, \tau_T)$. Dataset annotation assigns each step a binary error label $Y = (Y_1, \dots, Y_T) \in \{0, 1\}^T$, 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., 2026; Zheng et al., 2023; Zhao et al., 2026)) must be fixed before calibration. CROP controls risk relative to these annotations. We denote a labeled reasoning instance by $Z = (X, Y)$.

Step-Level Risk Proxy. For a reasoning instance X containing T steps, a vector-valued step-level risk proxy function s maps X to $s(X) = (s_1(X), \dots, s_T(X)) \in \mathbb{R}^T$, assigning one error-oriented score to each step. We write $R_{1:T} = s(X)$ and $R_t = s_t(X)$, where larger values indicate stronger evidence of an annotated error at

step τ_t . 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.

The proxy function need not be calibrated or perfectly accurate, but it must be fixed before conformal calibration. At the reasoning-instance level, labeled instances have three disjoint roles: proxy-training labels fit s when applicable, calibration labels select $\hat{\lambda}$, and test labels are used only for retrospective evaluation. At deployment, CROP computes $s(X)$ from the unlabeled instance X and requires no step-level error labels.

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 used to construct the step-level risk proxy function and all preprocessing and calibration-design choices fixed before observing the calibration instances, the labeled calibration reasoning instances $Z_1, \dots, Z_n$ and future labeled reasoning instance Z_{n+1} are exchangeable.*

Assumption 1 enforces reasoning-instance-level exchangeability after proxy training, preprocessing, and calibration-design choices have been fixed, but before the calibration instances are observed. Steps within a trace may be arbitrarily dependent. The assumption holds, for example, when calibration and future instances are sampled i.i.d. from the same task distribution, generated using the same fixed model and prompting procedure, and annotated under the same fixed protocol.

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 annotated erroneous step – can be represented as a binary loss that is monotonic with respect to the threshold. Because raising the threshold can only increase the number of retained steps, this loss can transition from non-failure to failure but not in the reverse direction. 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 $W_1, \dots, W_n, W_{n+1}$ 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*

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

whenever the feasible set is nonempty. If the feasible set is empty, set $\hat{\lambda} = \perp$, where $L_\perp(w) = 0$ for every w . Then

$$\mathbb{E}_{W_{1:n+1}} [L_{\hat{\lambda}}(W_{n+1})] \leq \alpha, \quad (2)$$

where the expectation is taken jointly over the exchangeable calibration sample $W_{1:n}$ and future example W_{n+1} . In particular, $\hat{\lambda}$ is random through its dependence on $W_{1:n}$.

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. $\square$

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:

$$M_\lambda(R_{1:T}) = \max \{ m \in \{0, \dots, T\} : R_t \leq \lambda \text{ for every } t \leq m \}. \quad (3)$$

Here, $M_\lambda(R_{1:T})$ is the number of initial reasoning steps retained at threshold λ . When the score vector is clear from context, we write $M_\lambda = M_\lambda(R_{1:T})$. 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:

$$L_\lambda(Z) = \mathbf{1}\{\exists t \leq M_\lambda(R_{1:T}) : Y_t = 1\}. \quad (4)$$

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

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

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

If $\mathcal{F}$ is nonempty, set $\hat{\lambda} = \max \mathcal{F}$; otherwise, set $\hat{\lambda} = \perp$, where $M_\perp(R_{1:T}) = 0$. 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). Set $\hat{\lambda} = \max \mathcal{F}$ when $\mathcal{F}$ is nonempty; otherwise, set $\hat{\lambda} = \perp$, with $M_\perp(R_{1:T}) = 0$. For the future labeled reasoning instance $Z_{n+1} = (X_{n+1}, Y_{n+1})$, evaluate $M_{\hat{\lambda}}$ on the score vector $s(X_{n+1})$. The returned prefix satisfies*

$$\mathbb{P}_{Z_{1:n+1}} \{ \exists t \leq M_{\hat{\lambda}} : Y_{n+1,t} = 1 \} \leq \alpha. \quad (6)$$

The probability is joint over calibration instances $Z_{1:n}$ and the future exchangeable instance Z_{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 error 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

$$\mathbb{E}_{Z_{1:n+1}} [L_{\hat{\lambda}}(Z_{n+1})] \leq \alpha,$$

where the expectation is joint over the calibration instances and the future exchangeable instance. Since this binary loss is the indicator of the event in Equation (6), its expectation is exactly the probability of prefix contamination. $\square$

Proposition 1 establishes finite-sample marginal risk control: over the random calibration sample and a future exchangeable instance, the probability that CROP returns a prefix containing an annotated error is at most α . This is not a conditional guarantee for each realized prefix.

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: CROP

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

- 1: **for** each calibration trace $i = 1, \dots, n$ **do**
- 2: $R_{i,1:T_i} \leftarrow s(X_i)$
- 3: **for** each threshold $\lambda \in \Lambda$ **do**
- 4: $M_{i,\lambda} \leftarrow \max\{m \in \{0, \dots, T_i\} : R_{i,t} \leq \lambda, \forall t \leq m\}$
- 5: $L_{i,\lambda} \leftarrow \mathbf{1}\{\exists t \leq M_{i,\lambda} : Y_{i,t} = 1\}$
- 6: **end for**
- 7: **end for**
- 8: $\mathcal{F} \leftarrow \left\{ \lambda \in \Lambda : \frac{1 + \sum_{i=1}^n L_{i,\lambda}}{n+1} \leq \alpha \right\}$
- 9: **if** $\mathcal{F} = \emptyset$ **then**
- 10: **return** empty-prefix fallback with zero contamination loss
- 11: **end if**
- 12: $\hat{\lambda} \leftarrow \max \mathcal{F}$
- 13: $R_{n+1,1:T_{n+1}} \leftarrow s(X_{n+1})$
- 14: $\hat{M} \leftarrow \max\{m \in \{0, \dots, T_{n+1}\} : R_{n+1,t} \leq \hat{\lambda}, \forall t \leq m\}$
- 15: **return** retained prefix $\tau_{n+1,1:\hat{M}}$

Algorithm 1 is verifier-agnostic and accepts any fixed step-level risk proxy function s , which may be based on a process reward model, learned detector, likelihood statistic, or hand-designed heuristic. The conformal guarantee requires reasoning-instance-level exchangeability and that s be fixed before calibration, but it does not require s to be intrinsically accurate or calibrated. More informative risk proxy functions can improve efficiency by retaining longer prefixes at the same target risk, while validity follows from the conformal calibration rule.

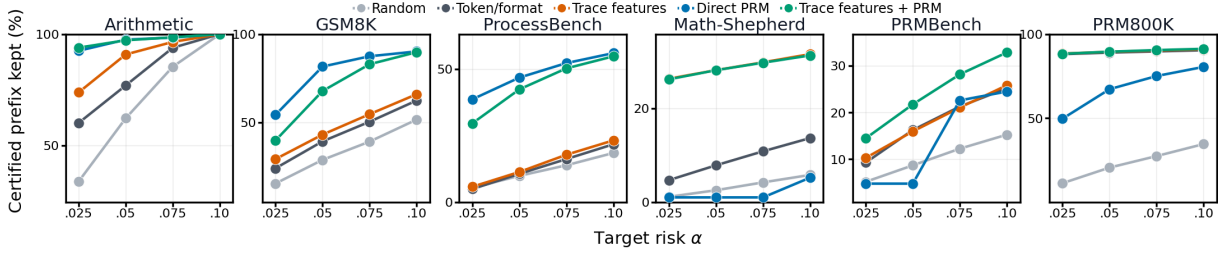


Figure 3: **CROP reveals how efficiently a risk proxy function converts risk budget into certified reasoning.** We swept the target prefix-contamination risk α over 10 random splits and reported the mean certified prefix retained after CROP calibration. The slope of each curve shows how much additional 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.

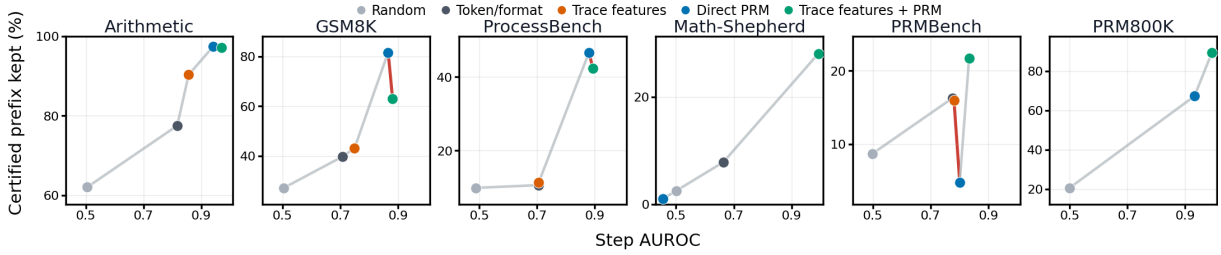


Figure 4: **Step AUROC is an incomplete indicator of fixed-risk prefix utility.** Each panel shows one dataset at $\alpha = 0.05$ over 10 random splits, with each point corresponding to one risk proxy type, averaged over splits; the x-axis is step-level AUROC and the y-axis is retained-prefix fraction. 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.

5 Experiments

We evaluated CROP across several step-level risk proxy functions, focusing on three questions: (1) Which risk proxy functions retain the longest prefixes at a fixed contamination-risk target? (2) How closely does the retained prefix approximate the oracle annotated-clean boundary? (3) Does supplying the retained prefix improve downstream repair?

Risk proxy functions. We evaluated five step-level scoring rules (Appendix B; scoring costs in Appendix Table 5):

1. *Random*: i.i.d. uniform step scores, drawn before conformal calibration as an uninformative reference.
2. *Token/format logistic*: logistic regression over surface features such as step position, length, and punctuation.
3. *Trace-feature logistic*: logistic regression combining token/format features with precomputed likelihood, confidence, and hidden-state summaries from Llama-3.1-8B-Instruct.

4. *Direct PRM*: the frozen Qwen2.5-Math-PRM-7B score, with error risk defined as one minus the positive-class probability.

5. *Trace features + PRM*: logistic regression combining the trace-feature inputs with the Direct PRM score. Its output is the predicted probability of the error class $Y_t = 1$.

For the three learned logistic models, we used an ℓ_2 penalty with $C = 1.0$ and the L-BFGS optimizer and fitted the model on the proxy-training partition using the local step label Y_t . The Direct PRM remained frozen and received no CROP labels.

Evaluation Metrics. For a target risk $\alpha = 0.05$, we evaluated retained prefixes on held-out test splits. For test trace i , let T_i denote its total number of steps and $M_i = M_{\hat{\lambda}}(R_{i,1:T_i})$ its retained prefix length. We define the oracle annotated-clean prefix length as

$$O_i = \min(\{t - 1 : Y_{i,t} = 1\} \cup \{T_i\}),$$

which is the number of steps before the first annotated error, or T_i when no annotated error exists. We report four metrics:

Table 1: **CROP improves boundary selection over whole-trace abstention at matched risk.** Both methods are separately calibrated at $\alpha = 0.05$ using the same risk proxy sources and trace-level splits. Values are mean percentages $\pm$ 95% confidence intervals across 20 splits. Bold marks higher retention and lower boundary components within each risk-matched pair; empirical risk is reported as a constraint and is not ranked. Question-only and full-trace references are excluded because they do not adapt to the requested risk budget. Results at additional risk targets are reported in Appendix E.

Dataset	Method	Empirical risk	Steps kept	Over-withholding	Unsafe overshoot	Boundary deviation
Arithmetic	Whole-trace abstention	4.4 ± 0.7	95.8 ± 0.7	2.4 ± 0.5	2.1 ± 0.3	4.5 ± 0.2
	CROP prefix	4.3 ± 0.4	97.2 ± 0.4	0.6 ± 0.2	1.7 ± 0.2	2.3 ± 0.3
GSM8K	Whole-trace abstention	5.0 ± 0.8	77.9 ± 2.0	13.4 ± 1.8	2.6 ± 0.4	15.9 ± 1.6
	CROP prefix	5.0 ± 0.8	81.6 ± 4.3	9.4 ± 4.2	2.3 ± 0.4	11.7 ± 4.0

1. *Retained-prefix fraction* (M_i/T_i): the fraction of original steps retained.
2. *Prefix-contamination risk*: the fraction of test traces whose retained prefix contains at least one annotated error.
3. *Boundary deviation* ($|M_i - O_i|/T_i$): the distance from the oracle boundary, decomposed into over-withholding $(O_i - M_i)_+/T_i$ and unsafe overshoot $(M_i - O_i)_+/T_i$.
4. *Repair accuracy*: normalized exact-match accuracy of the parsed final answer against the dataset gold answer.

Prefix-contamination risk evaluates validity relative to the target α . Among methods operating at comparable risk, higher retained-prefix fraction and lower boundary deviation indicate greater prefix utility, while higher repair accuracy indicates greater downstream utility.

Datasets. We evaluated CROP on six process-supervision datasets that span 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)¹. We also included Math-Shepherd (Wang et al., 2024a), Process-Bench (Zheng et al., 2025), PRMBench (Song et al., 2025), and PRM800K (Lightman et al., 2024).

¹We excluded a densely annotated Boolean domain (Zhao et al., 2026). Only 0.5% of its steps and 1.8% of its traces contain annotated errors. At $\alpha = 0.05$, domain-specific calibration retains 100% of the trace with 1.7% empirical prefix-contamination risk, producing a ceiling effect that does not meaningfully distinguish risk-proxy quality. With only approximately five erroneous traces per test split, the domain also cannot support an informative downstream repair comparison.

These supplementary datasets differ in trace construction and annotation semantics and do not provide uniform gold final-answer targets compatible with our repair evaluation. To maintain a consistent evaluation setup, we used them for fixed-risk prefix utility and restricted the boundary-quality and repair analyses to Arithmetic and GSM8K. All fixed-score evaluations used disjoint reasoning-instance-level 60/20/20 splits: 60% for fitting learned risk proxies, 20% for conformal calibration, and 20% for test-only evaluation. For fixed proxies that require no fitting, the calibration and test roles remain unchanged.

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 3 illustrates that score geometry influences 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 along with step-level AUROC at $\alpha = 0.05$ (Figure 4). 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.

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]

5.3 Boundary Quality of the Retained Prefix

Because retention alone does not distinguish unsafe overshoot from unnecessary withholding of annotated-clean steps, we compared CROP with conformal whole-trace abstention at $\alpha = 0.05$ (Table 1). Based on the preceding proxy comparison, we used PRM-backed proxies with strong fixed-risk prefix utility (Trace+PRM for Arithmetic and Direct PRM for GSM8K), fixed before the boundary and repair analyses without using repair outcomes. Within each dataset, both interfaces used the same proxy and were separately calibrated at the same α . At comparable empirical risk, CROP retained more steps and reduced over-withholding, unsafe overshoot, and total boundary deviation on both datasets. We report the corresponding risk-matched comparison at additional risk targets in Appendix E.

5.4 Certified Prefixes for Downstream Repair

Finally, we tested whether the retained prefixes improve 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. We report model details in Table 9 and qualitative results in Appendix F.2.

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 model-dependent: Llama yielded negative CROP deltas on both domains. One plausible contributor is same-family an-

swer repetition: the original traces were generated by a Llama-family model (Zhao et al., 2026), and our diagnostic shows that CROP reduced Llama’s repetition of the original wrong answer relative to full-trace input but not relative to question-only input (Appendix F.1). We provide a trace-level example in Table 11. These results indicate that a CROP prefix can help when removing a misleading suffix outweighs both lost context and contextual anchoring, but downstream gains remain model-specific.

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) at the primary risk target, CROP reduces boundary deviation relative to risk-matched whole-trace abstention; and (3) CROP improves downstream repair in the majority of the evaluated model–dataset settings. We discuss several further implications below.

PRM gains extend beyond superficial artifacts.

While simple surface cues (e.g., token and formatting features) provide useful baselines and indicate that the benchmark contains exploitable regularities, Table 16 demonstrates that PRM-backed scores remain substantially stronger even after reweighting examples by these cues and running label/order controls. Thus, these controls indicate that the PRM-backed gains are not explained solely

by the evaluated surface and dataset artifacts.

Boundary deviation at common risk budgets.

A fair boundary comparison must separately calibrate each interface to the same target risk. For $\alpha \in \{0.025, 0.05, 0.10\}$, CROP achieved lower boundary deviation than conformal whole-trace abstention in four of six dataset–risk settings and tied in one. At the primary $\alpha = 0.05$ setting, it retained more steps and reduced boundary deviation on both datasets (Appendix E). We exclude question-only and full-trace inputs from this comparison because they are fixed interfaces rather than risk-calibrated methods: question-only retains no trace context and therefore attains zero contamination trivially, whereas full trace generally exceeds the requested risk target. We retain both elsewhere as downstream utility references.

CROP changes the operational interface. Unlike whole-trace abstention, which retains an entire trace or falls back to the question, CROP returns a calibrated prefix and withholds any remaining uncertified suffix. At comparable empirical risk in the pooled Arithmetic and GSM8K evaluation, CROP retained more total step mass with PRM-backed scores despite accepting fewer traces in full, thereby recovering partial reasoning that whole-trace abstention could not retain (Table 7).

Boundary quality governs downstream repair.

Unlike whole-trace abstention, CROP can preserve useful intermediate reasoning while withholding an uncertified suffix. A retrospective analysis of originally incorrect traces with annotated errors supports this mechanism (Table 3). CROP improved repair when it stopped at or before the annotated error boundary, whereas these gains disappeared when it retained annotated-error steps. Thus, CROP is most useful when reusable initial reasoning is followed by a localized, misleading continuation.

Repair utility depends on the context consumer.

CROP controls annotated-prefix contamination risk, not the accuracy of an arbitrary downstream repair model. Nevertheless, at $\alpha = 0.05$, CROP improved over the best deployable non-CROP input in five of eight model–dataset settings, including five of six settings across Gemma, Qwen, and DeepSeek. The repetition diagnostic in Table 10 shows that Llama was more likely to reproduce the original wrong answer when supplied with trace context. This behavior is consistent with correlated failure modes between generation and repair (Kim

Table 3: **Repair gains depend on avoiding annotated-error overshoot.** Entries are CROP-minus-baseline repair-accuracy differences in percentage points on originally incorrect traces containing an annotated error. Before, At, and Beyond locate CROP relative to the oracle annotated-clean boundary, defined retrospectively from test labels. Values are means ± 1.96 standard errors over 20 splits; bold marks positive intervals excluding zero.

CROP relative to oracle	Question only	Full trace	Whole-trace abstention
Before	+5.7 $\pm$ 2.5	+21.9 $\pm$ 3.3	+9.0 $\pm$ 2.9
At	+5.9 $\pm$ 2.3	+20.3 $\pm$ 1.8	+8.7 $\pm$ 2.0
Beyond	−20.1 $\pm$ 3.0	−1.4 $\pm$ 2.3	−4.3 $\pm$ 2.4

et al., 2025; Tsui, 2025; Chen et al., 2026). Thus, downstream utility depends jointly on prefix quality and the repair model’s ability to use partial context.

Groupwise diagnostics. CROP provides marginal rather than group-conditional risk control, as is standard for marginal conformal guarantees. Our held-out diagnostics (Appendix H) show that the longest traces and harder problems tend to exhibit higher risk and lower retention, while dataset-level risk remains near the target after in-domain calibration despite variation in retained-prefix utility. Applications requiring group-specific control can use prespecified, deployment-observable groups for Mondrian (Vovk et al., 2003, 2005; Vovk, 2012) or conditional (Gibbs et al., 2025; Jung et al., 2023) calibration, provided that each group has sufficient calibration data.

Empty prefixes. The calibration-level fallback was never triggered in any runs across datasets, score functions, splits, and $\alpha \in \{0.025, 0.05, 0.075, 0.10\}$. At $\alpha = 0.05$, CROP nevertheless returned an empty prefix for 0% of Arithmetic and 6.4% of GSM8K test traces. A nonempty feasible set requires $1/(n+1) \leq \alpha$.

Prefix certification is complementary to conformal factuality.

Conformal factuality methods evaluate final answers, claim sets, or response specificity. CROP instead returns a contiguous initial portion of a reasoning trace with controlled annotated-error risk. These approaches are complementary: claim-level factuality scores, when defined at the reasoning-step level, could serve as risk proxies for CROP, while downstream factuality checks could evaluate a repaired continuation. CROP does not certify semantic truth or the causal faithfulness of written reasoning.

7 Limitations

CROP provides finite-sample marginal control of annotated prefix contamination over the calibration sample and a future exchangeable reasoning instance. The reasoning instance is the exchangeable unit, but the result is not a conditional guarantee that every returned prefix is correct with probability $1 - \alpha$. It also does not certify final-answer correctness, semantic truth, or causal faithfulness.

The practical interpretation depends on both distributional alignment and annotation quality. Obtaining reliable step-level labels could also be costly. Changes in the task, prompt distribution, generator, risk proxy, or annotation policy can violate exchangeability and require representative recalibration. Moreover, CROP controls agreement with the fixed annotation protocol rather than latent semantic correctness. Noisy, incomplete, or systematically biased annotations weaken the meaning of the certificate.

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 Yunpu Ma. 2025. Cot-kinetics: A theoretical modeling assessing llm 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.
- Kuan-Yen Chen, Fang-Yi Su, and Jung-Hsien Chiang. 2026. The self-correction illusion: Llms correct others but not themselves. *arXiv preprint arXiv:2606.05976*.
- 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*.
- Isaac Gibbs, John J Cherian, and Emmanuel J Candès. 2025. Conformal prediction with conditional guarantees. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 87(4):1100–1126.
- 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. 2026. A survey on llm-as-a-judge. *The Innovation*, 7(6).

- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shiron Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.
- Christopher Jung, Georgy Noarov, Ramya Ramalingam, and Aaron Roth. 2023. [Batch Multivalid Conformal Prediction](#). In *International Conference on Learning Representations*.
- 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*.
- Elliot Myunghoon Kim, Avi Garg, Kenny Peng, and Nikhil Garg. 2025. [Correlated errors in large language models](#). In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 30038–30066. PMLR.
- Klaus-Rudolf Kladny, Bernhard Schölkopf, and Michael Muehlebach. 2025. Conformal generative modeling with improved sample efficiency through sequential greedy filtering. In *International Conference on Learning Representations*, volume 2025, pages 78414–78443.
- 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 (UncertainNLP 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.
- Ken Tsui. 2025. Self-correction bench: Uncovering and addressing the self-correction blind spot in large language models. *arXiv preprint arXiv:2507.02778*.
- 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, Nicolo Dalmaso, 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. 2012. [Conditional validity of inductive conformal predictors](#). In *Proceedings of the Asian Conference on Machine Learning*, volume 25 of *Proceedings of Machine Learning Research*, pages 475–490, Singapore Management University, Singapore. PMLR.
- Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. 2005. *Algorithmic learning in a random world*. Springer.
- Vladimir Vovk, David Lindsay, Ilia Nouretdinov, and Alex Gammerman. 2003. Mondrian confidence machine. *Technical Report*.
- 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 full proof of Lemma 1. We condition on all training data used to fit the risk proxy function, as well as all preprocessing and calibration-design choices fixed before observing the calibration instances. Under this conditioning, the loss family $\{L_\lambda : \lambda \in \Lambda\}$ is fixed, and the calibration and future labeled reasoning instances remain exchangeable by Assumption 1. For CROP, each exchangeable example is a complete labeled reasoning instance $W_i = Z_i = (X_i, Y_i)$; steps within an instance may be arbitrarily dependent. We take $\alpha \in (0, 1)$ throughout.

Full proof of Lemma 1. Because $L_\lambda(w)$ is binary and nondecreasing in λ , its value can change from 0 to 1 at most once. For each example W_i , define its first failing threshold as

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

with $B_i = \infty$ if no threshold in Λ produces a failure. Therefore,

$$L_\lambda(W_i) = \mathbf{1}\{B_i \leq \lambda\}. \quad (8)$$

Since each B_i is computed from W_i using the same fixed loss family, exchangeability of $W_{1:n+1}$ implies exchangeability of $B_{1:n+1}$.

Let

$$k = \lfloor \alpha(n+1) \rfloor$$

and write the selected threshold explicitly as $\hat{\lambda} = \hat{\lambda}(W_{1:n})$. If the feasible set is empty, the procedure returns $\hat{\lambda} = \perp$ with $L_\perp(W_{n+1}) = 0$, so the result holds immediately. We therefore consider the case in which a non-fallback threshold is selected.

By the corrected calibration constraint,

$$1 + \sum_{i=1}^n L_{\hat{\lambda}}(W_i) = 1 + \sum_{i=1}^n \mathbf{1}\{B_i \leq \hat{\lambda}\} \quad (9)$$

$$\leq \alpha(n+1). \quad (10)$$

Because the left-hand side is integer-valued,

$$\sum_{i=1}^n \mathbf{1}\{B_i \leq \hat{\lambda}\} \leq k - 1. \quad (11)$$

Now suppose that the future example fails:

$$L_{\hat{\lambda}}(W_{n+1}) = 1.$$

Then $B_{n+1} \leq \hat{\lambda}$. Together with Equation (11), this implies

$$\sum_{i=1}^{n+1} \mathbf{1}\{B_i \leq \hat{\lambda}\} \leq k. \quad (12)$$

To handle ties, let $U_1, \dots, U_{n+1}$ be independent continuous random variables, independent of $W_{1:n+1}$, and rank the pairs (B_i, U_i) lexicographically. These pairs are exchangeable and almost surely have distinct ranks, so the rank of (B_{n+1}, U_{n+1}) is uniform on $\{1, \dots, n+1\}$. On the future failure event, Equation (12) implies that this rank is at most k . Consequently,

$$\mathbb{E}_{W_{1:n+1}} [L_{\hat{\lambda}}(W_{n+1})] = \mathbb{P}_{W_{1:n+1}} \quad (13)$$

$$\{L_{\hat{\lambda}}(W_{n+1}) = 1\} \quad (14)$$

$$\leq \frac{k}{n+1} \leq \alpha. \quad (15)$$

This proves the claim. $\square$

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), Process-Bench (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, . . . , 2825. For the additional-dataset fixed-risk experiments, we use 10 split seeds, 2806, . . . , 2815. The boundary-quality and risk-sensitivity analyses use the same 20 repeated target-domain splits.

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 $\Lambda = \{0, 0.01, \dots, 1.00\}$ 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 pre-computed 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 rescored 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,

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

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

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

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 $C = 1.0$, $\text{max_iter}=500$, $\text{class_weight}=\text{balanced}$, 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.

Comparison of annotation protocols. Table 4 shows that the datasets differ in both annotation source and granularity. The Arithmetic protocol combines programmatic checks with stronger-model judging, whereas GSM8K relies on the stronger-model judging pipeline. Both provide dense step labels, which allow us to derive exact first-error positions and conduct boundary and repair diagnostics, although the original dataset (Zhao et al., 2026) does not report a residual human-audited annotation-error rate. Process-Bench instead provides first-error/process annotations. This protocol is directly aligned with CROP calibration because the prefix-contamination loss requires only the first annotated error, or a no-error label; annotations after the first error do not affect threshold selection. Math-Shepherd supplies parsed step-markup labels, providing large-scale supervision under the semantics of its native markup. PRMBench provides fine-grained benchmark labels and represents a particularly stringent annotation regime, with most evaluated traces containing at least one annotated error. PRM800K supplies human process-supervision labels and therefore demonstrates compatibility with human rather than programmatic or model-judged annotations.

CROP is annotation-protocol agnostic in an operational, but not semantic, sense: it calibrates against any fixed protocol that identifies the first annotated error, but its guarantee inherits that protocol’s definition and errors. The resulting certificates therefore control agreement with each dataset’s native annotations; they do not establish that the protocols are equally accurate or interchangeable.

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.

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 rescore 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.

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 4 in the main text. We report the same fixed-risk evaluation numerically in Table 6, including the empirical prefix-contamination risk omitted from the 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 ev-

Table 6: **Fixed-risk prefix utility and step-ranking quality.** Entries report step AUROC, certified prefix retained (%), and empirical prefix-contamination risk (%) at $\alpha = 0.05$. Values are means $\pm$ 95% normal intervals over repeated trace-level splits. CRV target-domain results use 20 stratified splits; additional process-supervision results use 10 trace-level 60/20/20 splits. The intervals summarize variation across splits. Bold marks the best displayed non-random value within each dataset for AUROC and prefix retention separately; prefix risk is not ranked because it is the controlled constraint rather than the utility objective.

Dataset	Labels	Score source	Step AUROC	Prefix kept	Prefix risk
Arithmetic	dense step	Random	0.5049 $\pm$ 0.0164	62.0 $\pm$ 5.0	4.6 $\pm$ 0.6
		Token/format	0.8168 $\pm$ 0.0116	77.5 $\pm$ 2.7	4.7 $\pm$ 0.6
		Trace features	0.8564 $\pm$ 0.0153	90.3 $\pm$ 1.5	4.5 $\pm$ 0.6
		Direct PRM	0.9421 $\pm$ 0.0100	97.5 $\pm$ 0.4	4.2 $\pm$ 0.4
		Trace+PRM	0.9701 $\pm$ 0.0075	97.1 $\pm$ 0.4	4.1 $\pm$ 0.5
GSM8K	dense step	Random	0.5046 $\pm$ 0.0110	27.2 $\pm$ 2.6	4.3 $\pm$ 0.7
		Token/format	0.7081 $\pm$ 0.0126	39.8 $\pm$ 2.8	4.9 $\pm$ 0.7
		Trace features	0.7495 $\pm$ 0.0111	43.2 $\pm$ 2.7	4.3 $\pm$ 0.7
		Direct PRM	0.8659 $\pm$ 0.0088	81.7 $\pm$ 4.3	5.1 $\pm$ 0.8
		Trace+PRM	0.8812 $\pm$ 0.0113	63.1 $\pm$ 5.0	4.5 $\pm$ 0.8
ProcessBench	first-error/process	Random	0.4887 $\pm$ 0.0104	9.8 $\pm$ 1.6	4.9 $\pm$ 1.1
		Token/format	0.7057 $\pm$ 0.0034	10.6 $\pm$ 1.4	4.8 $\pm$ 0.5
		Trace features	0.7058 $\pm$ 0.0035	11.3 $\pm$ 1.0	4.5 $\pm$ 0.4
		Direct PRM	0.8801 $\pm$ 0.0059	46.7 $\pm$ 1.0	4.5 $\pm$ 0.6
		Trace+PRM	0.8935 $\pm$ 0.0081	42.4 $\pm$ 1.5	4.5 $\pm$ 0.5
Math-Shepherd	step markup	Random	0.5012 $\pm$ 0.0049	2.5 $\pm$ 0.2	4.3 $\pm$ 0.4
		Token/format	0.6637 $\pm$ 0.0024	7.8 $\pm$ 0.5	4.8 $\pm$ 0.5
		Trace features	0.9932 $\pm$ 0.0005	28.1 $\pm$ 0.3	5.2 $\pm$ 0.5
		Direct PRM	0.4537 $\pm$ 0.0043	1.0 $\pm$ 0.1	0.9 $\pm$ 0.1
		Trace+PRM	0.9944 $\pm$ 0.0004	28.1 $\pm$ 0.3	5.1 $\pm$ 0.5
PRMBench	fine-grained	Random	0.4994 $\pm$ 0.0050	8.7 $\pm$ 0.5	4.7 $\pm$ 0.4
		Token/format	0.7751 $\pm$ 0.0034	16.2 $\pm$ 1.0	5.0 $\pm$ 0.7
		Trace features	0.7812 $\pm$ 0.0034	15.9 $\pm$ 0.7	4.6 $\pm$ 0.5
		Direct PRM	0.8010 $\pm$ 0.0049	4.8 $\pm$ 0.2	0.6 $\pm$ 0.1
		Trace+PRM	0.8336 $\pm$ 0.0027	21.8 $\pm$ 0.9	5.0 $\pm$ 0.6
PRM800K	human step	Random	0.5002 $\pm$ 0.0100	20.5 $\pm$ 1.1	5.4 $\pm$ 0.5
		Token/format	0.9919 $\pm$ 0.0014	89.0 $\pm$ 0.3	4.8 $\pm$ 0.4
		Trace features	0.9925 $\pm$ 0.0013	89.4 $\pm$ 0.3	4.8 $\pm$ 0.4
		Direct PRM	0.9327 $\pm$ 0.0021	67.2 $\pm$ 1.2	4.5 $\pm$ 0.5
		Trace+PRM	0.9934 $\pm$ 0.0012	89.5 $\pm$ 0.4	4.6 $\pm$ 0.4

ery 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:

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

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

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:

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

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 $\lambda = 0.5$.

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

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

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

D Prefix Retention versus Whole-Trace Abstention

CROP and conformal whole-trace abstention control the same type of annotation-relative contamination risk but return different objects. Whole-trace abstention either retains every step or returns no trace context, whereas CROP returns a calibrated prefix and withholds any remaining uncertified suffix. We excluded Boolean before constructing the trace-level splits and separately calibrated both interfaces at $\alpha = 0.05$ on the pooled Arithmetic and GSM8K calibration traces using the same fixed score source.

Table 7 reports empirical contamination risk, step-weighted retention, and the fraction of traces retained in full. With Direct PRM and Trace+PRM scores, CROP retained respectively 4.3 and 3.5 percentage points more step mass despite retaining fewer traces in full. Token/format scores produced comparable retention. The low full-accept rate for Token/format CROP is consistent with these metrics: CROP rarely retained an entire trace but still preserved substantial partial reasoning.

E Boundary Sensitivity at Common Risk Budgets

We fixed $\alpha = 0.05$ before test evaluation and downstream repair; we did not select it to maximize retention, boundary quality, or repair accuracy. We additionally evaluated $\alpha \in \{0.025, 0.10\}$ using the same 20 trace-level 60/20/20 splits and domain-specific score sources as in the main experiment. At each target, we separately recalibrated CROP and conformal whole-trace abstention using the same score source. This comparison therefore holds the nominal risk budget fixed while allowing each interface to select its own calibrated threshold.

Table 7: **CROP exposes partial reasoning that whole-trace abstention cannot.** Results include Arithmetic and GSM8K only, with Boolean excluded before splitting. At target risk $\alpha = 0.05$, whole-trace abstention either retains or rejects the full trace, whereas CROP returns a calibrated prefix. *Kept* is the step-weighted percentage of reasoning retained, and *Full accept* is the percentage of traces retained in full. Values are means ± 1.96 standard errors across 20 repeated splits. Bold marks the higher retained-step mean within each score source; the paired retention interval excludes zero only for the PRM-backed scores.

Score	Object	Emp. risk	Kept	Full accept
Token/format	Whole	4.8 ± 0.5	59.8 ± 2.7	62.0 ± 2.4
	CROP	4.7 ± 0.5	60.5 ± 2.3	4.7 ± 2.8
Direct PRM	Whole	4.4 ± 0.5	85.6 ± 1.0	86.5 ± 0.9
	CROP	4.4 ± 0.4	89.8 ± 0.6	83.7 ± 0.8
Trace + PRM	Whole	4.5 ± 0.4	86.4 ± 1.0	87.4 ± 0.9
	CROP	4.6 ± 0.4	89.9 ± 0.6	82.2 ± 1.3

At the primary target $\alpha = 0.05$, paired split-level intervals support higher retention and lower boundary deviation for CROP on both datasets. Across all targets, CROP improved both quantities in four of six dataset–risk settings. Arithmetic saturated at $\alpha = 0.10$, producing an exact tie. On GSM8K at the stringent $\alpha = 0.025$ target, the point estimates favored whole-trace abstention, but the paired intervals included zero and therefore did not establish a reliable difference. These results indicate that the benefit of prefix certification depends on the operating point, while supporting the comparison at our pre-specified primary target.

We omit question-only and full-trace inputs from Table 8 because neither adapts to the requested risk budget. Question-only retains no reasoning and therefore has zero contamination, whereas full trace always retains every step and generally violates stringent targets. Including these fixed interfaces in a risk-matched ranking would conflate descriptive input references with methods that are separately calibrated to α .

F 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

Table 8: **CROP versus whole-trace abstention at common risk budgets.** Absolute results are mean percentages $\pm$ 95% normal intervals over 20 trace-level splits. Paired differences (Δ) are CROP minus abstention, with 95% intervals in brackets. Positive retained-prefix differences and negative boundary-deviation differences favor CROP. Bold marks favorable paired differences whose intervals exclude zero. The intervals quantify sensitivity across overlapping splits rather than independent-sample population uncertainty.

Dataset	α	Interface	Empirical risk	Steps kept	Boundary deviation
Arithmetic	0.025	Whole-trace abstention	1.9 ± 0.4	90.9 ± 1.2	7.1 ± 1.0
		CROP	2.0 ± 0.4	93.9 ± 0.9	3.9 ± 0.8
		Δ (CROP – abstention)	–	+3.0 [2.2, 3.8]	–3.2 [–3.9, –2.6]
	0.050	Whole-trace abstention	4.4 ± 0.7	95.8 ± 0.7	4.5 ± 0.2
		CROP	4.3 ± 0.4	97.2 ± 0.4	2.3 ± 0.3
		Δ (CROP – abstention)	–	+1.4 [0.8, 2.1]	–2.2 [–2.5, –1.9]
	0.100	Whole-trace abstention	8.3 ± 0.0	100.0 ± 0.0	3.9 ± 0.1
		CROP	8.3 ± 0.0	100.0 ± 0.0	3.9 ± 0.1
		Δ (CROP – abstention)	–	$0.0 [0.0, 0.0]$	$0.0 [0.0, 0.0]$
GSM8K	0.025	Whole-trace abstention	2.1 ± 0.6	56.4 ± 6.6	34.3 ± 6.2
		CROP	2.2 ± 0.7	54.2 ± 7.5	36.4 ± 7.1
		Δ (CROP – abstention)	–	$–2.2 [–6.4, 2.0]$	$+2.1 [–2.0, 6.2]$
	0.050	Whole-trace abstention	5.1 ± 0.8	77.9 ± 2.0	15.9 ± 1.6
		CROP	5.1 ± 0.8	81.6 ± 4.3	11.7 ± 4.0
		Δ (CROP – abstention)	–	+3.7 [0.6, 6.9]	–4.2 [–7.0, –1.5]
	0.100	Whole-trace abstention	9.1 ± 0.9	86.6 ± 1.1	11.6 ± 0.5
		CROP	9.2 ± 0.8	90.2 ± 0.7	7.1 ± 0.4
		Δ (CROP – abstention)	–	+3.5 [2.6, 4.5]	–4.5 [–4.9, –4.1]

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 artifacts. We report the model family, Ollama tag and access date in Table 9. We used deterministic decoding for all four repair models. For each model and split, all input modes used the same checkpoint, prompt construction, and decoding configuration.

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

Table 9: **Downstream repair model artifacts.** We report the Ollama model tag requested by each repair experiment and the date on which it was accessed for the recorded run.

Model family	Ollama tag	Access date
Gemma 4 E4B IT	gemma4:latest	2026-05-11
Qwen2.5-7B-Instruct	qwen2.5:7b-instruct	2026-05-13
DeepSeek-R1-0528-Qwen3-8B	deepseek-r1:8b	2026-05-14
Llama-3.1-8B-Instruct	llama3.1:8b	2026-05-13

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 $-\backslash d+(?:\backslash \backslash \backslash d+)?$ 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.

F.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 10 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.

Table 11 provides a concrete case in which question-only repair succeeds, but Llama reproduces the original wrong answer when given trace context. This occurs even though CROP withholds both the erroneous calculation and the wrong final answer.

F.2 Qualitative CROP and Repair Examples

We present four held-out examples, two from each repair dataset, illustrating two mechanisms through which CROP can improve measured repair accuracy: removing a misleading suffix and retaining useful intermediate computations. Each table shows the complete step sequence, native step annotations, CROP boundary, and actual repair outputs. The examples use the calibrated CROP threshold associated with their respective trace-level split.

G 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

Table 10: **Llama3.1-8B-Instruct same-family answer-repetition diagnostic.** Cells report mean percentages $\pm$ 95% normal intervals 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. Bold marks the best point estimate within each domain and metric. Because test partitions overlap across splits, the intervals quantify split sensitivity rather than independent-sample population uncertainty.

Domain	Input mode	Repeat original wrong answer	Recover original wrong answer	Degrade original correct answer
Arithmetic	Question only	12.8 $\pm$ 3.0	47.4 $\pm$ 4.1	24.1 $\pm$ 0.9
	Full trace	74.3 $\pm$ 3.8	9.6 $\pm$ 2.7	6.2 $\pm$ 0.4
	Whole-trace abstention	47.2 $\pm$ 6.0	24.0 $\pm$ 3.0	3.9 $\pm$ 0.4
	CROP prefix	31.8 $\pm$ 4.8	23.5 $\pm$ 4.7	5.1 $\pm$ 0.5
GSM8K	Question only	8.1 $\pm$ 1.4	47.8 $\pm$ 2.6	21.5 $\pm$ 1.3
	Full trace	31.1 $\pm$ 2.3	42.7 $\pm$ 2.4	17.0 $\pm$ 1.3
	Whole-trace abstention	9.3 $\pm$ 1.4	55.4 $\pm$ 1.9	19.3 $\pm$ 1.0
	CROP prefix	11.6 $\pm$ 1.6	53.1 $\pm$ 1.7	19.5 $\pm$ 0.8

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 breaks the relationship between labels and traces during risk proxy fitting, and the other disrupts the step order within traces. Table 16 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.

H Groupwise Diagnostics

We evaluated held-out prefix contamination and retained-prefix utility by trace length, problem difficulty, dataset, and first-annotated-error position at the primary target $\alpha = 0.05$. We designed these analyses as diagnostics of heterogeneity rather than as new group-conditional certificates. For every split, we fit the risk proxy on the score-training partition, selected the CROP threshold on the calibration partition, and computed subgroup metrics only on the held-out test partition. We did not recal-

ibrate thresholds within the diagnostic groups. Consequently, the CROP guarantee remains marginal within each calibrated dataset or domain rather than a simultaneous or group-conditional guarantee.

Diagnostic protocol. We used the same reasoning-instance-level split protocol and fixed score choices as in the main experiments. For Arithmetic and GSM8K, we averaged results over the same 20 repeated 60/20/20 trace-level splits used in the repair analysis. We used Trace features + PRM for Arithmetic and Direct PRM for GSM8K, calibrated separate thresholds within the two domains, and then pooled their held-out traces for the trace-length and first-error-position summaries. For the dataset analysis, we retained those domain-specific choices and used the selected fixed score for each external dataset: Direct PRM for ProcessBench and Trace features + PRM for Math-Shepherd, PRMBench, and PRM800K. We averaged the external-dataset results over 10 repeated trace-level splits.

We formed four trace-length bins: 2-5, 6-7, 8-9, and 10 or more steps. For problem difficulty, we used the native CRV Arithmetic complexity levels 3, 5, 7, and 10; GSM8K and the external datasets do not provide this field, so we did not include them in that diagnostic. For first-error position, we divided the one-indexed relative location of the first annotated error by trace length and formed early (0-33%), middle (34-66%), and late (67-100%) bins. We computed this last diagnostic only among traces containing an annotated error. Because first-error position is annotation-derived and unavailable when CROP is deployed, we use it only for retrospective analysis.

Table 11: **Trace-level example of same-family answer repetition.** CROP retains all 14 annotated-clean steps and withholds the two annotated-error steps. Nevertheless, Llama3.1-8B reconstructs the same erroneous computation and repeats the original wrong answer.

Step	Original generated trace	Annotation	CROP
Problem. Becca, Smendrick, and PJ have 341 cards in total. Becca has 12 more cards than Smendrick, and Smendrick has three times as many as PJ. How many cards does Becca have? Gold answer: 153. Original answer: 147.			
1	Let’s break this problem down step by step.	Clean	Retained
2	Let’s say PJ has x cards.	Clean	Retained
3	Smendrick has $3x$ cards.	Clean	Retained
4	Becca has $3x + 12$ cards.	Clean	Retained
5	The total number of cards is 341, so we set up the equation:	Clean	Retained
6	$x + 3x + (3x + 12) = 341$.	Clean	Retained
7	Combine like terms.	Clean	Retained
8	$7x + 12 = 341$.	Clean	Retained
9	Subtract 12 from both sides.	Clean	Retained
10	$7x = 329$.	Clean	Retained
11	Divide both sides by 7.	Clean	Retained
12	$x = 47$.	Clean	Retained
13	PJ has 47 cards.	Clean	Retained
14	Smendrick has $3x = 3(47) = 141$ cards.	Clean	Retained
15	Becca has $3x + 12 = 3(47) + 12 = 147$ cards.	Error	Withheld
16	Therefore, Becca has 147 cards.	Error	Withheld
Llama3.1-8B repair outputs			
Question only	“ $x = 47 \dots$ Smendrick has 141 cards. Becca has 12 more than Smendrick, so Becca has 153 cards. FINAL: 153”		Correct
Full trace	“Becca has $3(47) + 12 = 147$ cards. FINAL: 147”		Repeats 147
CROP prefix	“Becca has $3x + 12 = 3(47) + 12 = 147$ cards. FINAL: 147”		Repeats 147
Interpretation. The CROP prompt contains neither the erroneous calculation in step 15 nor the wrong final answer in step 16. The repair model nevertheless reconstructs the same invalid equality, $3(47) + 12 = 147$, whereas it computes the correct answer from the question alone. Because the erroneous calculation and answer are absent from the CROP prompt, this behavior cannot be explained by directly copying the withheld suffix and is consistent with model-specific contextual repetition.			

For each group, we computed empirical prefix-contamination risk as the fraction of held-out traces for which the returned prefix included at least one annotated error. We computed retained-prefix utility as the mean returned-prefix length divided by full trace length. We report means and 95% normal-approximation intervals over split seeds. Because test partitions overlap across repeated seeds, these intervals quantify sensitivity to the train/calibration/test split rather than independent-sample uncertainty or confidence intervals for a population parameter.

Trace length. We observed stable empirical risk near or below the marginal target for traces containing two through nine steps: risk increased from 2.8% for 2-5 steps to 4.8% for both 6-7 and 8-9 steps, while retention remained between 90.7% and 91.7%. For traces with at least 10 steps, risk increased to 9.7% and retention decreased to 81.8%. Long traces were also the smallest length group, with approximately 60 test traces

per split, and their interval was wider than those of the shorter groups. The pattern nevertheless indicates meaningful length-dependent heterogeneity: longer traces provide more opportunities for a score to miss an error before the stopping boundary and contain a higher held-out error prevalence (27.4% in the raw diagnostic, compared with approximately 11.3-13.5% in the shorter bins). We therefore do not interpret the pooled marginal certificate as conditional control for long traces; deployments requiring length-specific control should prespecify length groups and calibrate them with sufficient groupwise data.

Problem difficulty. We found a monotone deterioration across the upper Arithmetic complexity levels. Risk was 2.3% and 2.2% at complexity levels 3 and 5, increased to 5.6% at level 7, and reached 7.3% at level 10. Retention remained high throughout but declined from approximately 99% at levels 3 and 5 to 96.2% at level 7 and 94.7% at level 10. The held-out group sizes were balanced at

Table 12: **Arithmetic suffix-removal example (arith.nt10.annotated:1355)**. The complete step sequence is shown with native CRV labels and the CROP boundary; step text is lightly typeset without changing its calculations. DeepSeek-R1-8B is the repair model.

Step	Original generated trace	Annotation	CROP
Problem. $((6 + 0)(6(6 \cdot 6))) + (((1 + 3) + 4) + (4 \cdot 2)) \cdot 3$. Gold: 1344. Original: 20784.			
1	To evaluate this expression, we need to follow the order of operations (PEMDAS).	Clean	Retained
2	Evaluate the innermost expressions: $6 \cdot 6 = 36$, $6 \cdot 36 = 216$, $6 + 0 = 6$, $6 \cdot 216 = 1296$, $1 + 3 = 4$, $4 + 4 = 8$, $4 \cdot 2 = 8$, and $8 + 8 = 16$.	Clean	Retained
3	Multiply 1296 by 16: $1296 \cdot 16 = 20736$.	Error	Withheld
4	Multiply 16 by 3: $16 \cdot 3 = 48$.	Error	Withheld
5	Add 20736 and 48: $20736 + 48 = 20784$.	Error	Withheld
6	The final answer is 20784.	Error	Withheld
Repair outputs			
Question only	Computes the left term as 1296 but reaches the 192-token generation limit while beginning the right term; parsed answer 1296.		Wrong
Full trace	Correctly reconstructs the two subexpressions as 1296 and 48 but reaches the 192-token limit after “1296+”; parsed answer 1296.		Wrong
CROP prefix	“The final result is $1296 + (16 \times 3) = 1296 + 48 = 1344$.”		Correct
Why. CROP stops exactly before the first annotated error and removes the spurious $1296 \cdot 16$ operation. Its clean prefix elicits a concise completion that correctly combines 1296 and 48 within the fixed generation budget. Because the question-only and full-trace outputs reach that budget, this case illustrates both suffix removal and a generation-length effect.			

approximately 74-76 traces per split, making gross sample-size imbalance an unlikely explanation for this trend. The raw diagnostic instead shows that annotated-error prevalence rose from 3.5-5.0% at levels 3 and 5 to 11.3% and 13.6% at levels 7 and 10. Thus, the common Arithmetic threshold retained nearly all reasoning at every level, but it did not equalize contamination risk across difficulty.

Dataset and annotation protocol. We observed substantially more stable empirical risk across datasets than across trace length or problem difficulty. Despite differences among dense hybrid annotations, stronger-model judgments, first-error annotations, parsed step markup, fine-grained benchmark labels, and human process supervision, mean empirical risk ranged only from 4.3% to 5.1% across the six datasets. This result shows that in-domain CROP calibration can provide similar annotation-relative risk control under each dataset’s native labeling interface. It does not establish that the annotations have equal semantic accuracy or noise rates.

Retained-prefix utility varied much more substantially, from 21.8% on PRMBench and 28.1% on Math-Shepherd to 97.2% on Arithmetic and 89.5% on PRM800K. ProcessBench achieved 4.5% risk with 46.7% retention using first-error/process labels, demonstrating that dense labels are not required for CROP calibration. At the other extreme, PRMBench contains an annotated

error in 96.4% of held-out traces; CROP therefore retained only 21.8% while maintaining 5.0% empirical risk. PRM800K, which uses human process labels and has substantially lower annotated-error prevalence, attained 4.6% risk with 89.5% retention. These comparisons illustrate the distinction between validity and efficiency: CROP can attain similar risk relative to each protocol while returning very different amounts of reasoning.

We do not attribute these retention differences causally to annotation protocol alone. Dataset difficulty, trace origin, error prevalence, trace structure, risk-proxy choice, and score separability also vary across datasets.

First annotated error position. Among erroneous traces, we found contamination risks of 29.9%, 38.1%, and 29.8% for early, middle, and late first errors, respectively. These values do not violate the 5% marginal guarantee because we conditioned on the outcome-defining subset of traces that contain an annotated error; every clean trace, which necessarily has zero prefix-contamination loss, is excluded from this diagnostic. We also grouped traces using the first annotated error itself, which is unavailable at deployment and was not used for calibration. We found that retained-prefix utility increased from 30.6% for early errors to 51.3% for middle errors and 63.2% for late errors. This increase is operationally desirable because later errors leave more clean reasoning available before

Table 13: **Arithmetic retained-context example (arith.nt10.annotated:1580)**. The complete step sequence is shown with native CRV labels and the CROP boundary; step text is lightly typeset without changing its calculations. Gemma 4 8B is the repair model.

Step	Original generated trace	Annotation	CROP
Problem. $-((1 + 9) \cdot 4) + (6 + 9)((5 + 2) \cdot 5) + 0$. Gold: 485. Original: 565.			
1	To evaluate this expression, we need to follow the order of operations (PEMDAS).	Clean	Retained
2	Evaluate the innermost expressions: $(1 + 9) = 10$ and $(5 + 2) = 7$.	Clean	Retained
3	Multiply the results: $10 \cdot 4 = 40$ and $7 \cdot 5 = 35$.	Clean	Retained
4	Add zero to the results: $40 + 0 = 40$ and $35 + 0 = 35$.	Clean	Retained
5	Evaluate $(6 + 9) = 15$ and multiply $15 \cdot 35 = 525$.	Clean	Retained
6	Add the results: $40 + 525 = 565$.	Error	Withheld
7	Add zero: $565 + 0 = 565$.	Error	Withheld
8	The final answer is 565.	Error	Withheld
Repair outputs			
Question only	Computes 40, 35, and 525 but reaches the 160-token limit before combining the signed outer terms; parsed answer 525.		Wrong
Full trace	“Simplify the first main term: -40 [and] the second main term: 525. Combine the terms: $-40 + 525 = 485$.”		Correct
CROP prefix	“This becomes $-40 + (525 + 0)$. Finally: $-40 + 525 = 485$. FINAL: 485”		Correct
Why. CROP retains all five annotated-clean setup steps and removes the suffix that drops the outer negation. The retained values expose the required combination $-40 + 525$, although the question-only contrast is partly confounded by output truncation.			

the oracle boundary. However, the non-monotone conditional contamination pattern, with the highest value for middle errors, shows that the fixed threshold does not localize every first error uniformly. The early-error group was small at approximately 12 traces per split and consequently had the widest split-sensitivity interval.

Overall interpretation. We draw three conclusions from these diagnostics. First, calibrating within dataset or domain kept dataset-level empirical risk close to 5% across all six datasets, despite large differences in retained-prefix utility. Second, marginal calibration did not imply conditional calibration: the longest traces and hardest Arithmetic problems showed elevated contamination risk, identifying concrete settings where prespecified groupwise calibration or additional calibration data may be useful. Third, the first-error-position analysis explains where utility arises but cannot serve as a deployable grouping rule because it uses the unknown error annotation. These findings preserve the paper’s marginal guarantee while making its scope explicit: CROP controls average annotation-relative prefix contamination under exchangeability, but users who require subgroup-specific control must define deployment-available groups before calibration and ensure adequate calibration size in each group.

I Distribution-Shift Diagnostic

Cross-domain transfer. We evaluated how CROP behaves when the calibration and deployment distributions do not match. We intentionally fit the risk proxy and calibrated the CROP threshold on one target domain, then applied the resulting rule to the other domain. This experiment violates the reasoning-instance exchangeability assumption, so we present it as a failure diagnostic rather than a conformal validity result.

Table 17 shows a substantial asymmetric failure under cross-domain transfer. When we trained and calibrated on Arithmetic and evaluated on GSM8K, empirical prefix-contamination risk increased to 14.8%, substantially exceeding the nominal 5% target. When we trained and calibrated on GSM8K and evaluated on Arithmetic, empirical risk was 5.1%. Although the latter value is close to the target in this particular experiment, it carries no conformal guarantee because the exchangeability assumption is still violated.

This diagnostic demonstrates that successful calibration on one domain does not automatically transfer to another. Changes to the task, prompt distribution, generator, risk proxy, verifier, or annotation policy can alter the joint distribution of risk scores and annotated errors. CROP therefore requires representative recalibration when the deployment distribution changes. Prespecified groupwise cali-

Table 14: **GSM8K suffix-removal example (gsm8k.annotated:1146)**. The complete step sequence is shown with native CRV labels and the CROP boundary; step text is lightly typeset without changing its calculations. DeepSeek-R1-8B is the repair model.

Step	Original generated trace	Annotation	CROP
Problem. Adam invests \$100 in rocks costing \$5 each, sells them for \$7 each, and sells only 60% of his inventory. How much money does he lose? Gold: 16. Original: 56.			
1	Adam can buy $100/5 = 20$ rocks.	Clean	Retained
2	Selling all 20 rocks would yield $20 \cdot 7 = \$140$.	Clean	Retained
3	He sells $0.6 \cdot 20 = 12$ rocks and receives $12 \cdot 7 = \$84$.	Clean	Retained
4	He loses $140 - 84 = \$56$ because he could have sold all the rocks.	Error	Withheld
5	The final answer is 56 .	Error	Withheld
Repair outputs			
Question only	Computes the relevant loss as $100 - 84 = 16$ but continues discussing unsold inventory until reaching the 192-token limit; parsed answer 8.		Wrong
Full trace	Repeats the trace’s comparison with hypothetical full-sale revenue: “ $140 - 84 = 56$. FINAL: 56”		Wrong
CROP prefix	“The initial investment is 100, so the loss is $100 - 84 = 16$. FINAL: 16”		Correct
Why. CROP stops exactly before the first annotated error. The retained prefix contains the purchase cost and realized revenue, while withholding the incorrect comparison against hypothetical full-sale revenue. This directly supports the suffix-removal mechanism, although the question-only output is also affected by truncation.			

bration can address known heterogeneity defined by deployment-observable groups, but it is not a general remedy for arbitrary distribution shift.

J 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.

K 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.

L Information About Use of AI Assistants

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

Table 15: **GSM8K retained-context example (gsm8k.annotated:912)**. The complete step sequence is shown with native CRV labels and the CROP boundary; step text is lightly typeset without changing its calculations. DeepSeek-R1-8B is the repair model.

Step	Original generated trace	Annotation	CROP
Problem. John drinks one bottle every 30 minutes. A normal Sudoku takes 45 minutes and an extreme Sudoku takes four times as long. How many bottles does he drink during the extreme Sudoku? Gold: 6. Original: 4.5.			
1	A normal Sudoku takes 45 minutes.	Clean	Retained
2	An extreme Sudoku takes $4 \cdot 45 = 180$ minutes.	Clean	Retained
3	The time difference is $180 - 45 = 135$ minutes.	Error	Withheld
4	There are $135/30 = 4.5$ half-hour intervals in the difference.	Error	Withheld
5	John therefore drinks 4.5 bottles.	Error	Withheld
6	The trace discusses rounding or interpreting 4.5 bottles.	Error	Withheld
7	The final answer is $\boxed{4.5}$.	Error	Withheld
Repair outputs			
Question only	Correctly computes $180/30 = 6$ in its reasoning but reaches the 192-token limit after continuing with an alternative hourly calculation; parsed answer 2.		Wrong
Full trace	Identifies that the trace uses the time difference rather than total duration and returns $\boxed{6}$.		Correct
CROP prefix	Uses the retained 180-minute duration: “ $180/30 = 6$. FINAL: 6”		Correct
Why. CROP stops exactly before the first annotated error and retains the key intermediate duration needed to finish the calculation. The CROP result is consistent with useful retained context, but the question-only contrast is also affected by output truncation.			

Table 16: **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	84.0 ± 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

Table 17: **Cross-domain failures illustrate why calibration must match deployment.** Here CROP is calibrated on one domain and evaluated on another, so the conformal exchangeability assumption is intentionally broken. This is not a validity claim. Prefix risk, marginal false acceptance (FA), and accept are trace-weighted percentages; prefix kept is the trace-weighted mean of per-trace retained-prefix percentages.

Train/cal domain	Test domain	Prefix risk	Prefix kept	FA	Accept
Arithmetic	GSM8K	14.8	69.9	13.3	62.5
GSM8K	Arithmetic	5.1	69.0	4.4	54.7

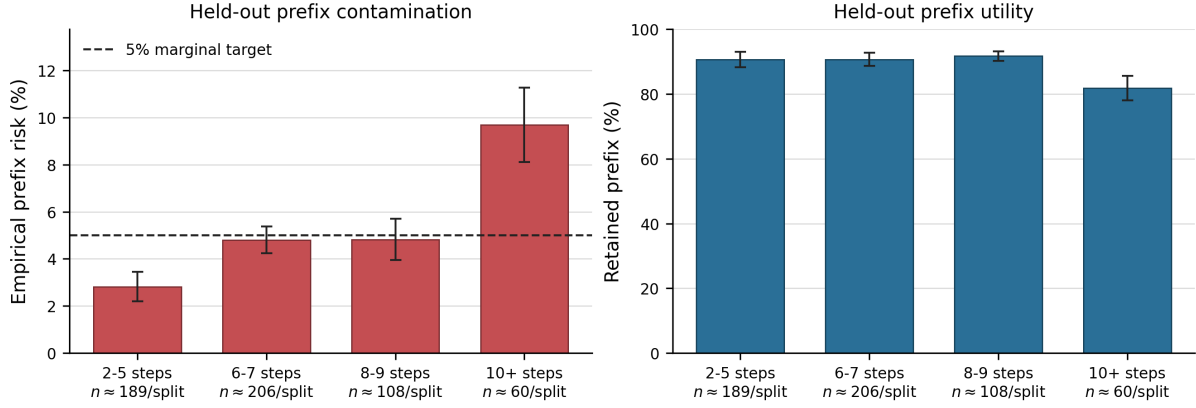


Figure 5: **Groupwise diagnostics reveal elevated risk for long traces.** We pool Arithmetic and GSM8K only after applying their separately calibrated domain-specific thresholds. We report empirical prefix-contamination risk (left) and retained-prefix fraction (right), with 95% intervals over 20 split seeds and the mean held-out group size per split below each label. Empirical prefix-contamination risk remains near or below the marginal 5% target for traces with 2–9 steps, while traces with 10 or more steps reach 9.7% risk and retain 81.8% of their steps. The dashed line is a marginal reference, not a length-conditional guarantee.

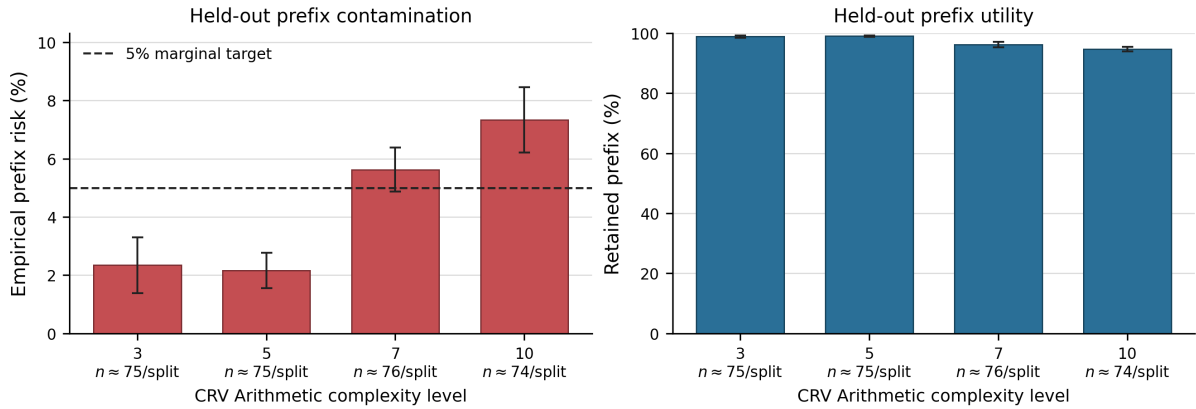


Figure 6: **Prefix risk increases with Arithmetic problem difficulty.** We report empirical prefix-contamination risk (left) and retained-prefix fraction (right), with 95% intervals over 20 split seeds. We use the fixed Arithmetic Trace features + PRM protocol and do not recalibrate within complexity levels. Empirical risk rises from 2.2–2.3% at complexity levels 3–5 to 5.6% at level 7 and 7.3% at level 10, while retained-prefix utility remains above 94%.

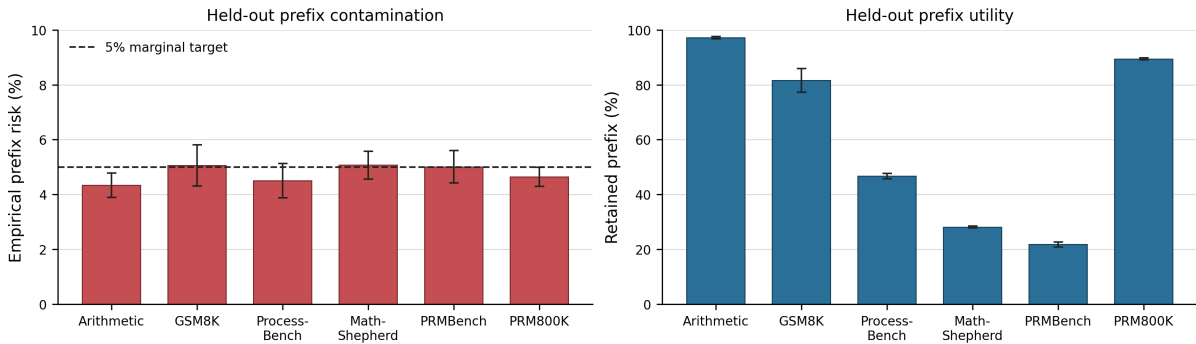


Figure 7: **Dataset-level risk remains near the target, but retained-prefix utility varies substantially.** We report empirical prefix-contamination risk (left) and retained-prefix fraction (right), with 95% intervals over repeated trace-level splits. We use 20 splits for Arithmetic and GSM8K and 10 for the external datasets, and we calibrate within each dataset or target domain using its fixed score protocol. Empirical risk ranges from 4.3% to 5.1% across all six datasets, whereas retained-prefix utility ranges from 21.8% on PRMBench to 97.2% on Arithmetic.

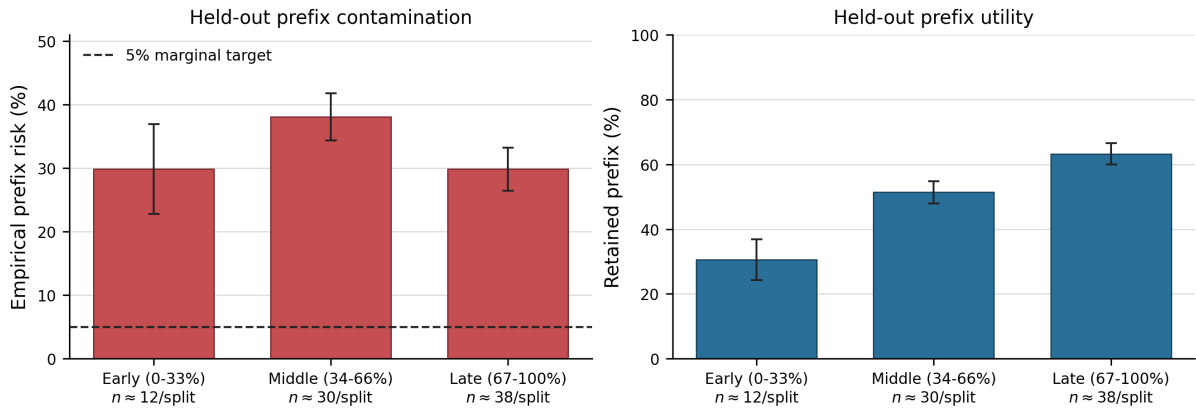


Figure 8: **Later first errors leave more reusable reasoning, but conditional contamination remains heterogeneous.** We report empirical prefix-contamination risk among erroneous traces (left) and retained-prefix fraction (right), with 95% intervals over 20 split seeds. Because we exclude all clean traces and define groups using an annotation unavailable at deployment, the dashed marginal target is a reference only and does not apply conditionally to these bars. Among erroneous traces, retained-prefix utility increases from 30.6% for early errors to 63.2% for late errors, while conditional contamination ranges from 29.8% to 38.1%.