CERTIFIED PROCESS VERIFICATION: TRAJECTORY-LEVEL RISK CONTROL FOR LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Process verifiers make local decisions, but deployment failures can be compositional: a trajectory is contaminated if any failure-relevant error is certified as safe. We study this mismatch between the *decision unit* (step) and the *risk unit* (trajectory). **Certified Process Verification (CPV)** preserves fine-grained step-level certify/flag decisions while calibrating the trajectory-level loss jointly induced by those decisions. For nonmonotone selective losses CPV uses Learn-then-Test (LTT), while monotone trajectory contamination can be calibrated with either LTT or conformal risk control (CRC), depending on the desired guarantee semantics. On AgentProcessBench (APB), simple trajectory-aware alternatives control contamination but are substantially more conservative: union-bound and independence-based length corrections certify 0.0% and 1.2% of steps, max-score trajectory aggregation certifies 6.3%, and CROP certifies 9.1%, whereas CPV LTT-HB certifies 14.6% at trajectory risk 0.051 ± 0.013 for a target of 0.1. An oracle-score decomposition raises CPV certification to 38.6%, showing that both finite-sample calibration and scorer quality limit efficiency. Across 120 matched independent synthetic calibration draws under the same trajectory-contamination loss, LTT-HB incurs $0/120$ high-probability violations with mean risk 0.0360, while CRC attains mean risk 0.0862 under its expected-risk guarantee, consistent with the distinct semantics of the two procedures. Results across APB, Who&When, and DeltaBench characterize trajectory-safe process verification as a validity–utility problem in which fine-grained decisions and trajectory-level risk must be jointly calibrated under a common deployment objective.

1 INTRODUCTION

Large language model (LLM) agents solve tasks through sequences of reasoning, tool calls, code execution, retrieval, and intermediate decisions. Errors can therefore arise before the final output and propagate through later steps. Process reward models, step verifiers, and LLM judges address this problem by scoring intermediate behavior (Wang et al., 2024; Zhang et al., 2025b; Khalifa et al., 2026; Zhao et al., 2026; Chae et al., 2025; Xia et al., 2025), while recent benchmarks explicitly annotate process errors and, in some settings, decisive failure points (Zhang et al., 2025a; Fan et al., 2026; He et al., 2025; Zheng et al., 2025) across diverse agent trajectories in deployed settings.

Deployment, however, requires more than ranking erroneous steps. A verifier must decide which steps can be certified and which require additional review. Metrics such as AUROC and AUPRC characterize score quality but do not control the probability that an erroneous step is certified as safe. More importantly, the relevant deployment event may be whether *any* failure-relevant error passes within an entire trajectory, rather than whether a randomly selected step is misclassified when viewed in isolation outside its full trajectory context and the preceding deployment decisions.

Figure 1 illustrates the resulting mismatch. Let trajectory i contain steps $x_{i1}, \ldots, x_{iT_i}$, with $e_{it} = 1$ indicating a failure-relevant error under the benchmark annotation semantics, and let $A_\lambda(x_{it}) = 1$ denote certification under threshold λ. We define trajectory contamination as

$$L_i^{\mathrm{traj}}(\lambda) = \mathbf{1}\left[\exists\, t : e_{it} = 1 \wedge A_\lambda(x_{it}) = 1\right]. \tag{1}$$

A verifier may therefore make decisions locally while risk is defined by the union of those decisions over a complete trajectory. The key distinction is *decision unit = step* but *risk and calibration*

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

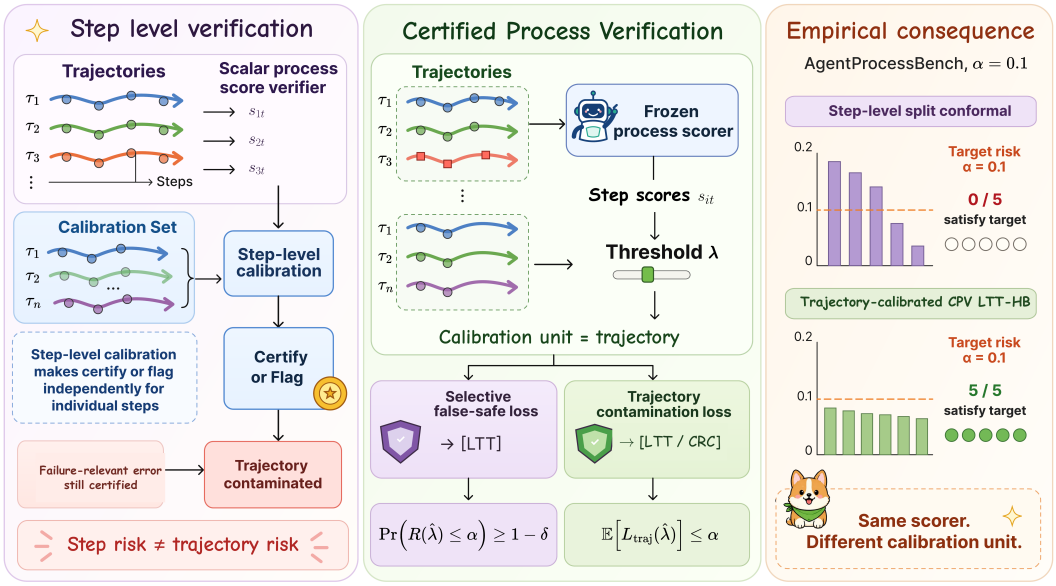


Figure 1: Overview of Certified Process Verification. Existing verifiers score and certify individual steps, while the relevant deployment failure may be induced jointly over the complete trajectory. CPV preserves step-level certify/flag decisions but uses the trajectory as the calibration and risk unit. The right panel summarizes the matched APB comparison reported in Table 1 under the same risk target in deployment.

unit = trajectory. CPV does not collapse the output into a single trajectory label; it calibrates the trajectory-level loss induced by fine-grained step decisions over complete deployment trajectories.

We introduce **Certified Process Verification (CPV)**, a post-hoc calibration framework for arbitrary frozen process scorers. CPV constructs one bounded loss per complete trajectory and then applies the calibration machinery appropriate to the loss geometry. Selective false-safe ratios are generally nonmonotone and therefore use Learn-then-Test (LTT) (Angelopoulos et al., 2025). Trajectory contamination is monotone, so CRC (Angelopoulos et al., 2024) supplies expected-risk control, while LTT supplies a high-probability (α, δ) statement when that guarantee is desired. The statistical contribution is the formulation of deployment losses that align local certification decisions with the global trajectory-level event they jointly induce across complete deployment trajectories.

This framing also changes the empirical question: can trajectory-aware calibration control the target while retaining useful step-level certification? On APB at $\alpha = \delta = 0.1$, a union-bound correction certifies no steps and an independence-based correction certifies 1.2%; max-score trajectory aggregation certifies 6.3% and CROP (Cheung et al., 2026) 9.1%, whereas CPV LTT-HB certifies 14.6% at risk 0.051 ± 0.013. An oracle-score run with the same CPV calibration reaches 38.6%, separating finite-sample conservatism from scorer limitations. Synthetic repeated-calibration experiments provide direct validation of the stated guarantee semantics under repeated sampling.

Our contributions are:

1. We identify a statistical-unit mismatch in process verification: verifiers score and certify individual steps, while deployment failure can depend on whether any failure-relevant error is certified anywhere in the complete trajectory during deployment in practice.
2. We formulate the trajectory-level risks induced by step certification. Selective false-safe risk captures the composition of certified steps and is generally nonmonotone, whereas trajectory contamination captures whether any failure-relevant error passes and is monotone in the certification threshold used throughout deployment.
3. We introduce CPV, a model-agnostic framework that preserves fine-grained step-level certify/flag decisions while calibrating their induced trajectory losses. LTT handles nonmonotone losses with high-probability (α, δ) control; CRC controls expected monotone contamination without assuming independent within-trajectory steps.
4. Through matched trajectory-aware baselines, oracle-score decomposition, three process-verification benchmarks, and 120 independent synthetic calibration draws, we characterize

the validity–utility tradeoff of trajectory-safe process verification and show that CPV is substantially less conservative than simple length corrections and coarse trajectory aggregation without collapsing the underlying step-level decisions at deployment time.

2 Related Work

Process verification. Process reward models and step verifiers score intermediate reasoning, tool use, and agent actions (Wang et al., 2024; Zhang et al., 2025b; Khalifa et al., 2026; Zhao et al., 2026; Chae et al., 2025; Xia et al., 2025). ProcessBench, AgentProcessBench, DeltaBench, and Who&When further provide step-level annotations of process errors and decisive failure points (Zheng et al., 2025; Fan et al., 2026; He et al., 2025; Zhang et al., 2025a). These advances make it possible to identify where an agent trajectory becomes unreliable. CPV builds on this capability by turning process scores into deployment decisions: it determines which individual steps can be certified while controlling the risk induced over the complete trajectory. This connects process-error detection to risk-controlled deployment rather than treating verification only as a ranking or localization problem in isolation during deployment under an explicit operational risk target.

Conformal and selective risk control. Conformal prediction and risk-controlling prediction sets provide distribution-free tools for converting predictive scores into decisions with finite-sample guarantees under exchangeability (Bates et al., 2021; Angelopoulos et al., 2025; 2024). Related work has extended selective prediction and risk control more broadly (Xu et al., 2025), including to language models and agent decision making (Khosravi & Huo, 2026; Badshah et al., 2026). CPV brings this perspective to process verification, where local certification decisions jointly determine a trajectory-level failure event. The resulting problem has a characteristic separation between the decision unit and the risk unit: decisions are made at individual steps, while risk is evaluated on the complete trajectory. CPV therefore constructs trajectory-wise losses induced by step-level certification decisions. Selective false-safe risk is generally nonmonotone and is calibrated with Learn-then-Test, whereas trajectory contamination is monotone and supports conformal risk control. This formulation provides finite-sample risk control without sacrificing fine-grained step-level decisions during downstream deployment with explicit guarantees for the selected trajectory-level loss.

Representation-based error detection. Frozen LLM representations encode useful signals about correctness, hallucination, and uncertainty (Orgad et al., 2025; Azaria & Mitchell, 2023). We use a lightweight probe over frozen representations as an efficient process scorer within CPV. The calibration layer operates on scalar scores and therefore also supports process reward models, generative judges, and other verifier architectures. This separation allows improvements in process scoring to translate directly into higher certification efficiency while leaving the risk-control mechanism unchanged across the supported scorer families and deployment settings considered here.

3 Certified Process Verification

3.1 Problem Setup

Let trajectory i contain the following ordered sequence of steps:

$$\tau_i = (x_{i1}, \ldots, x_{iT_i}).$$

Each judgeable step has an error indicator $e_{it} \in \{0, 1\}$ and a mask $m_{it} \in \{0, 1\}$ indicating whether the corresponding step has a released judgeable annotation. For trajectory contamination, we write $e_{it}^{\text{caus}} \in \{0, 1\}$ for the benchmark-specific indicator of a failure-relevant error. Its construction follows each benchmark's released annotation semantics and is specified for each dataset in Appendix A. Missing or nonjudgeable annotations are excluded rather than imputed, preserving the released label semantics throughout calibration and evaluation.

A frozen process scorer g maps each step to a scalar score

$$s_{it} = g(x_{it}) \in \mathbb{R}, \tag{2}$$

where smaller values indicate safer steps. The score may come from a process reward model, a generative judge, or the frozen-representation probe described in Section 3.5. CPV does not require s_{it} to be calibrated as a probability before it is used by CPV for threshold selection.

For threshold λ, the deployment rule is defined by the following stepwise decision function:

$$A_\lambda(x_{it}) = \mathbf{1}[s_{it} \leq \lambda]\,, \tag{3}$$

where $A_\lambda(x_{it}) = 1$ denotes certification and $A_\lambda(x_{it}) = 0$ denotes flagging for additional review. Increasing λ therefore certifies a weakly larger set of steps while retaining their stepwise outputs.

The central statistical choice in CPV is to use complete trajectories as the calibration observations. Let $\tau_1, \ldots, \tau_n$ denote the calibration trajectories. Steps within one trajectory may be arbitrarily dependent; the finite-sample guarantees require exchangeability across trajectories and a future test trajectory, rather than independence among constituent steps within any individual trajectory.

3.2 Selective False-Safe Risk

For an error indicator e_{it}, define the trajectory-wise false-safe ratio

$$L_i^{\mathrm{sel}}(\lambda) = \frac{\sum_t m_{it} e_{it} A_\lambda(x_{it})}{\max\{1, \sum_t m_{it} A_\lambda(x_{it})\}}. \tag{4}$$

Its population risk is $R_{\mathrm{sel}}(\lambda) = \mathbb{E}[L_i^{\mathrm{sel}}(\lambda)]$. The indicator e_{it} may represent any annotated process error, including a benchmark-defined failure-relevant error under its released annotation rules.

Because increasing λ can add both correct and erroneous certified steps, this ratio need not be monotone. We therefore calibrate candidate thresholds with LTT rather than a monotone conformal rule. This objective is relevant when the composition of the certified set is itself the deployment target; the main experiments below focus on trajectory contamination, which directly captures whether any failure-relevant error is certified anywhere in the trajectory during deployment.

3.3 Trajectory Contamination Risk

Many deployments have a stricter operational requirement: a trajectory is unsafe if *any* failure-relevant error is certified. Let $e_{it}^{\mathrm{caus}} = 1$ denote a step designated as failure-relevant under the benchmark-specific annotation semantics. We define the corresponding trajectory-contamination loss as

$$L_i^{\mathrm{traj}}(\lambda) = \mathbf{1}\left[\exists\, t : e_{it}^{\mathrm{caus}} = 1 \wedge A_\lambda(x_{it}) = 1\right], \tag{5}$$

with the associated population risk given by

$$R_{\mathrm{traj}}(\lambda) = \mathbb{E}\Big[L_i^{\mathrm{traj}}(\lambda)\Big]\,. \tag{6}$$

This is the probability that a new trajectory contains at least one failure-relevant error that the verifier certifies as safe. Unlike the selective ratio, $L_i^{\mathrm{traj}}(\lambda)$ is monotone nondecreasing in λ because the certified set expands monotonically with increasing thresholds.

Why the calibration unit matters. Even under independent step-wise false-safe events with probability q, a trajectory of length T is contaminated according to the following closed-form probability:

$$1 - (1 - q)^T, \tag{7}$$

which exceeds q whenever $T > 1$. Thus, controlling a per-step error rate at level α does not generally control trajectory contamination at the same level. CPV instead calibrates the union event directly using one loss per complete trajectory without requiring independent within-trajectory steps.

3.4 Finite-Sample Calibration

Learn then Test. Let Λ be a finite grid of thresholds. For each $\lambda \in \Lambda$, CPV computes the empirical trajectory risk

$$\widehat{R}_n(\lambda) = \frac{1}{n} \sum_{i=1}^{n} L_i(\lambda)$$

and tests $H_\lambda : R(\lambda) > \alpha$ using a valid finite-sample Hoeffding–Bentkus or exact-binomial p-value. After Bonferroni correction over Λ, CPV selects the certified threshold with the largest certified-step rate. Full p-value formulas and proofs are given in Appendix F.

Theorem 1 (Trajectory-level LTT validity)**.** *Suppose the calibration trajectories and a future test trajectory are exchangeable. If the tests of $H_\lambda : R(\lambda) > \alpha$ control familywise error at level δ, then the selected threshold $\lambda^\star$ satisfies*

$$\mathbb{P}(R(\lambda^\star) \leq \alpha) \geq 1 - \delta.$$

No independence assumption is required among steps within one trajectory for the stated result.

Conformal Risk Control. For monotone trajectory contamination, CRC selects

$$\widehat{\lambda}_{\mathrm{CRC}} = \sup\left\{\lambda : \frac{n}{n+1}\widehat{R}_n(\lambda) + \frac{1}{n+1} \leq \alpha\right\}. \tag{8}$$

Under exchangeability,

$$\mathbb{E}\left[L^{\mathrm{traj}}_{\mathrm{test}}(\widehat{\lambda}_{\mathrm{CRC}})\right] \leq \alpha.$$

LTT provides a high-probability (α, δ) statement and also applies to nonmonotone losses; CRC provides expected-risk control for monotone losses under exchangeable trajectories.

3.5 Frozen-Representation Process Scorer

As a lightweight default scorer, we extract final-layer hidden states from a frozen instruction-tuned LLM, fit PCA on the training split, and train an ℓ_2-regularized logistic probe with class-balanced weights. The backbone remains frozen, and representations are cached before calibration. CPV can therefore change α, δ, or the calibration method without rerunning the frozen language model.

This scorer is only one CPV instantiation. Better scorers can improve certification efficiency, while the calibration layer supplies the risk criterion independently of the scorer architecture in practice.

4 Experiments

Our experiments ask whether trajectory-level calibration changes deployment risk, whether useful certification remains possible across risk budgets, and whether the framework generalizes across scorers and benchmarks under the matched evaluation protocol used throughout.

4.1 Experimental Setup

Datasets and splits. We evaluate on the full released datasets: AgentProcessBench (APB; 1,000 trajectories, 8,509 step annotations) (Fan et al., 2026), Who&When (184 failed trajectories with decisive-step labels) (Zhang et al., 2025a), and DeltaBench (1,236 long reasoning trajectories) (He et al., 2025). APB and DeltaBench use trajectory-disjoint $50/25/25$ train/calibration/test splits; the main APB results average five seeded splits. Who&When uses five seeded evaluation runs. Synthetic trajectories provide an additional guarantee check under a known data-generating process with repeated, independently sampled calibration draws.

Scorers, baselines, and metrics. Our default scorer is a frozen Qwen2.5-7B representation probe (Qwen et al., 2024). We additionally evaluate alternative frozen backbones, PRMs, and an LLM judge. Besides step-level calibration, we include trajectory-aware alternatives that allocate the target risk across trajectory length, aggregate step scores into a trajectory-level max score, or certify a structured prefix (CROP). We compare methods at the same trajectory-risk target and report trajectory contamination, step certification, scorer AUROC/AUPRC, first-error localization, and fixed-budget review utility. An oracle-score diagnostic keeps the CPV calibration procedure fixed while replacing the learned ranking signal. Full preprocessing and hyperparameters are given in Appendix A.

4.2 Matched Trajectory-Aware Calibration

We first compare methods under the same trajectory-contamination target $\alpha = \delta = 0.1$. This comparison separates the need for trajectory-aware control from the utility of the calibration rule used in this controlled deployment comparison across all matched baselines.

Trajectory-aware alternatives expose the central validity–utility tradeoff. The union-bound rule certifies no steps and the independence correction certifies only 1.2%. Max-score trajectory aggregation

Table 1: Trajectory-aware calibration on AgentProcessBench at $\alpha = \delta = 0.1$ (mean±std over five seeded splits). "Sat." is an empirical diagnostic counting held-out splits with test risk at or below α; it is not the formal LTT coverage statement for any single split or a simultaneous guarantee across all five splits.

Method	Traj. risk ↓	Step cert. ↑	Sat./5
Step split conformal	0.169±0.030	**0.327±0.015**	0/5
Union bound (α/T_i)	**0.000±0.000**	0.000±0.000	5/5
Independence correction	0.028±0.010	0.012±0.005	5/5
Max-score trajectory aggregation	0.087±0.018	0.063±0.012	5/5
CROP (structured prefix) (Cheung et al., 2026)	0.035±0.020	0.091±0.014	5/5
CRC (trajectory) (Angelopoulos et al., 2024)	0.091±0.017	0.219±0.023	4/5
CPV LTT-HB	0.051±0.013	0.146±0.011	5/5

Table 2: Oracle-score decomposition on APB at $\alpha = \delta = 0.1$. The oracle score is a diagnostic and is not deployable in practical verification because it uses ground-truth labels at scoring time during the diagnostic.

Setting	Step cert. rate ↑
Oracle upper bound	**0.671**
Oracle score + CPV LTT-HB	0.386
Learned score + CPV LTT-HB	0.146

raises certification to 6.3%, and CROP to 9.1%. CPV LTT-HB certifies 14.6% while retaining trajectory risk 0.051 ± 0.013. Thus, trajectory awareness alone is not sufficient for useful verification: the main advantage of CPV is to preserve fine-grained step decisions while calibrating the global event they induce over complete trajectories during deployment.

CRC is reported separately in guarantee semantics. Its $4/5$ empirical count does not contradict its expected-risk guarantee; individual splits may exceed the nominal target even when expected risk is controlled. We return to this distinction in Section 4.7 with matched synthetic evidence.

Where does the remaining conservatism come from? Table 2 keeps the CPV LTT-HB calibration pipeline fixed and replaces the learned score with an oracle ranking signal. The unconstrained oracle upper bound certifies 67.1% of steps, oracle scores followed by the same finite-sample CPV calibration certify 38.6%, and the learned scorer certifies 14.6%. The two gaps show that both finite-sample calibration and scorer quality materially limit certification efficiency.

4.3 Risk and Certification Tradeoffs

Using trajectory-level CRC, we vary the trajectory-risk budget to examine how the calibrated decision rule moves along the validity–utility frontier. Across $\alpha \in \{0.05, 0.10, 0.20\}$, certification increases from 9.3% through 21.9% to 30.6%, while empirical trajectory contamination is 0.033, 0.091, and 0.177, respectively, below the corresponding targets. The increase is monotone in the allowed risk budget and shows that CPV exposes an operational control knob rather than only a post-hoc error estimate for policy selection.

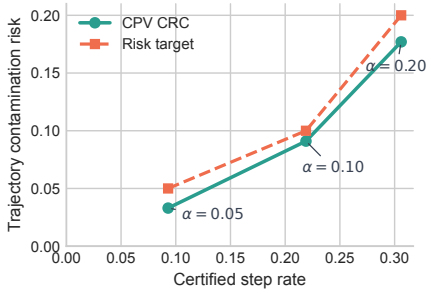


Figure 2: Risk–certification tradeoff. Larger risk budgets permit higher certification under the corresponding trajectory-risk target.

The frontier also complements the fixed-α comparison in Table 1. Table 1 asks which trajectory-aware procedure retains useful step-level decisions at the same target, whereas Figure 2 asks how one calibrated procedure changes as the deployment tolerance itself varies. Together they characterize both the choice of calibration rule and the choice of risk budget available to downstream deployment operators under their selected target in practice.

4.4 Generality Across Scorers and Benchmarks

CPV separates process-score quality from calibration validity. We first compare the frozen-representation scorer with existing verifier families under matched evaluation conditions.

The Qwen2.5-7B frozen probe reaches AUROC 0.721 on the principal APB split and 0.756 ± 0.030 across five seeds while requiring only a lightweight head over cached representations. Figure 3

Table 3: Process-error scoring on AgentProcessBench. CPV probes use the principal split and a lightweight supervised head with frozen backbones; external-verifier numbers retain their released evaluation subsets and reported precision. Results therefore compare usable scorer signals under their respective evaluation and supervision settings. Values are displayed to three decimal places; source values released to two decimal places are zero-padded for alignment rather than treated as additional precision in this table or analysis.

Verifier	AUROC ↑	AUPRC ↑	FirstErr ↑
Qwen2.5 Math PRM 7B (Zhang et al., 2025b)	0.635	0.540	0.273
Qwen2.5 32B judge (Qwen et al., 2024)	0.568	0.426	0.303
ThinkPRM 14B (Khalifa et al., 2026)	0.660	0.550	0.310
GenPRM 32B (Zhao et al., 2026)	0.680	0.570	0.330
AgentRM 8B (Xia et al., 2025)	0.670	0.560	<u>0.340</u>
Web Shepherd 8B (Chae et al., 2025)	0.640	0.530	0.300
CPV probe, 7B	<u>0.721</u>	<u>0.624</u>	0.326
CPV probe, Qwen3 8B	0.707	0.619	0.303
CPV probe, 32B	**0.812**	**0.709**	**0.411**

connects ranking quality to deployment utility: stronger frozen backbones generally permit a less conservative threshold under the same trajectory-risk target and calibration method.

This pattern is consistent with the oracle decomposition in Table 2. Calibration determines which thresholds can be certified, while the scorer determines how many safe steps lie below those thresholds. Improving the scorer therefore translates directly into higher usable certification without changing the CPV risk-control layer or its formal guarantee across the evaluated scorer variants.

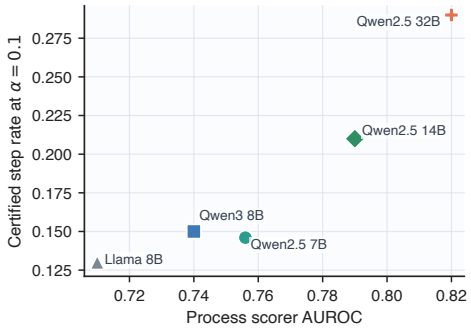


Figure 3: Scorer quality vs. certification. Stronger frozen representations support higher certification at $\alpha = 0.1$ with LTT-HB. Backbones: Qwen2.5, Qwen3, and Llama 3 (Qwen et al., 2024; Yang et al., 2025; AI@Meta, 2024). Five-seed aggregates are shown; Table 3 reports the principal calibration split used here for reference.

On Who&When, the same frozen-representation approach also localizes the annotated decisive failure point. The Qwen3-8B probe (Yang et al., 2025) reaches 0.185 ± 0.051 decisive-step localization accuracy across five seeds, above the reported 0.142 benchmark result (Zhang et al., 2025a). This result confirms that the inexpensive scorer captures meaningful failure-localization signal in addition to supporting calibrated certification under the same frozen-representation scoring framework and decision rule.

The cross-benchmark results also expose an important small-sample effect. Table 4 reports trajectory risk and certification on Who&When, where each seeded calibration split contains 36 or 37 trajectories, alongside the five-seed APB aggregate and the principal DeltaBench result. CPV remains nonvacuous on Who&When but is substantially more conservative: step certification is 6.8%, and the fraction of fully certified trajectories ranges from 0.0% to 5.4% across the five seeds. These results provide both step- and trajectory-level views of the efficiency loss induced by limited calibration data. Appendix A details the seeded splits for exact replication.

Table 4: Cross-benchmark trajectory control at $\alpha = 0.1$. Who&When provides a small-sample calibration stress test under limited data, with only 36–37 calibration trajectories per seeded split in each run.

Dataset	Trajectory risk	Step cert.
APB	0.051	0.146
Who&When	0.000	0.068
DeltaBench	0.036	0.071

Across datasets, the same trajectory-level formulation remains applicable, but certification efficiency varies with both calibration-set size and scorer quality. The Who&When result therefore illustrates how limited calibration data can substantially reduce trajectory-controlled certification efficiency.

Taken together, these results separate two sources of usable verification: the calibration procedure determines which decisions can be certified at the requested trajectory-risk level, while the scorer determines how efficiently safe and unsafe steps can be ranked within that constraint. This distinc-

tion also motivates a second operational question: how useful are the remaining scores when review capacity is limited in realistic deployments and budgets are tight?

4.5 Review Efficiency

The same process score can prioritize limited review resources. At a 10% review budget, score-guided review identifies 25.1% of contaminating errors compared with 10.4% under random allocation; at a 20% budget the corresponding rates are 44.4% and 19.0%. The same scalar score therefore supports two complementary deployment actions: low-risk steps can be certified, while high-risk steps can be prioritized for additional scrutiny within the same operational pipeline throughout deployment.

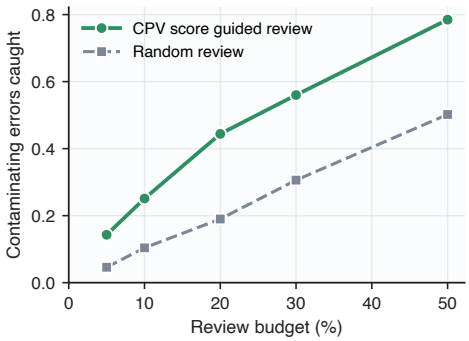


Figure 4: Review efficiency. Score-guided review consistently outperforms random review at every evaluated budget in deployment tests.

The gain persists across all evaluated review budgets. This matters because certification rate alone does not describe the value of the remaining flagged steps: even when CPV does not certify a step, the score still orders review effort toward the errors most likely to contaminate a trajectory. The review experiment therefore provides a second utility view of the same process score and complements certification performance across review budgets.

4.6 Ablation Studies

Table 5 summarizes the main design choices at $\alpha = \delta = 0.1$ under a common evaluation target.

Table 5: Key CPV ablations on AgentProcessBench at $\alpha = \delta = 0.1$. "Sat." counts seeded splits with empirical test risk at or below α. The calibration-size rows are a dedicated resampling ablation, separate from the default $50/25/25$ split. Certified rates are displayed to three decimal places; overlapping rows reproduce the corresponding five-seed means from Table 1. Bold and underline mark the best and second-best certification rates within each axis; for two-setting axes, only the better rate is bolded.

Axis	Setting	Certified ↑	Sat./5
Calibration unit	Step split conformal	**0.327**	0/5
	Trajectory LTT-HB	0.146	5/5
Finite-sample test	Hoeffding UCB	0.000	5/5
	Exact binomial	**0.149**	5/5
	LTT-HB	<u>0.146</u>	5/5
Calibration method	CRC (Angelopoulos et al., 2024)	**0.219**	4/5
	CROP (Cheung et al., 2026)	0.091	5/5
	Mondrian (Vovk et al., 2003)	0.030	5/5
	LTT-HB	<u>0.146</u>	5/5
Calibration size	$n_{\text{cal}} = 75$	0.101	5/5
	$n_{\text{cal}} = 295$	**0.151**	5/5
Score features	Logistic probe	0.146	5/5
	Execution fusion	**0.179**	5/5

The ablations separate validity from the main sources of conservatism within trajectory-aware verification. Changing the calibration unit alters the risk object being controlled, while the finite-sample test, calibration size, and score features primarily change how much certification can be retained once trajectory-level validity is enforced. Simple Hoeffding is especially conservative, whereas exact-binomial and Hoeffding–Bentkus retain 14.9% and 14.6% certification, respectively, under identical scoring conditions and a matched target with the same frozen scorer.

The alternative calibration procedures further expose the efficiency–conservatism tradeoff. CRC is more aggressive under its distinct expected-risk semantics, whereas CROP and Mondrian are more conservative and LTT-HB provides high-probability control with intermediate certification. The "Sat." counts remain descriptive diagnostics rather than a common inferential score across these procedures. Increasing n_{cal} from 75 to 295 raises certification from 10.1% to 15.1%; separately, execution-feature fusion raises certification from 14.6% to 17.9%.

Table 6: Matched synthetic validation over 120 independent calibration draws. Both procedures act on the same monotone trajectory-contamination loss and use the same certification-efficiency definition. "Draws $> \alpha$" is the LTT violation count; for CRC it is only a diagnostic because CRC controls expected risk rather than each calibration draw under repeated sampling of the same procedure across the matched repetitions.

Method	Target	Draws $> \alpha$	Mean risk $\downarrow$	Max risk $\downarrow$	Mean cert. $\uparrow$
LTT-HB	$\alpha = \delta = 0.1$	0/120	**0.0360**	**0.0803**	0.0119
CRC	$\alpha = 0.1$	15/120	0.0862	0.1348	**0.0291**

4.7 Finite-Sample Guarantee Validation

We validate the finite-sample guarantee directly in a synthetic setting with known population risk. We therefore evaluate LTT and CRC on the *same* synthetic trajectory-contamination loss, using the same 120 independent calibration draws, the same score orientation and threshold grid, and the same large test pool for population-risk evaluation. Table 6 reports this matched guarantee check.

Under this matched trajectory-loss evaluation, LTT-HB is more conservative than CRC, as expected from its stronger high-probability guarantee: it attains both lower mean trajectory risk (0.0360 versus 0.0862) and lower mean step certification (0.0119 versus 0.0291). LTT-HB incurs no population-risk violations among the 120 independent draws. CRC has 15 individual draws above 0.1, but its mean risk remains below the target at 0.0862; this does not contradict CRC because its guarantee is on expected risk rather than a high-probability statement for every calibration draw. The matched experiment therefore validates the distinct semantics of the two procedures without mixing selective-risk and trajectory-contamination experiments. Additional synthetic details are reported in Appendix B for completeness and independent verification of both guarantee semantics.

5 Discussion

Guarantee and efficiency. CPV controls the deployment loss defined by the chosen annotation semantics. For trajectory contamination, one loss is constructed per complete trajectory, so no independence assumption is required among steps within the same trajectory. The finite-sample guarantees instead rely on exchangeability across calibration and future trajectories. The matched baselines show that trajectory awareness by itself can be highly conservative, while the oracle-score decomposition shows that CPV's remaining efficiency gap reflects both finite-sample calibration and scorer quality. Validity and utility should therefore be reported jointly in deployment.

Assumptions and distribution shift. Generator shift, task shift, or changes in trajectory composition can violate the exchangeability assumption underlying the main guarantees. We therefore treat weighted calibration and online adaptation as empirical extensions rather than part of the primary finite-sample result. Additional shift and adaptation experiments are reported in the appendix.

Scope of certification. CPV controls risk relative to the benchmark's definition of error and cannot resolve systematic annotation mistakes or unobserved failure modes. It also does not prescribe what happens after a step is flagged. Human review, stronger verification, regeneration, or repair can all be placed downstream; CPV provides the risk-controlled certify-or-review decision that precedes these interventions in downstream deployment and operational handling in practice.

6 Conclusion

Process verification combines local decisions with a global deployment event. CPV preserves step-level certify/flag decisions while calibrating the trajectory-level loss they jointly induce. Matched trajectory-aware baselines show that simple length corrections, whole-trajectory aggregation, and structured prefix certification can control contamination but at substantially lower step-certification utility. An oracle-score decomposition quantifies how finite-sample calibration and scorer quality each contribute to the remaining conservatism, and 120 independent synthetic calibration draws directly validate the distinct LTT and CRC guarantee semantics. Trajectory-safe process verification therefore requires aligning the statistical unit, decision granularity, and guarantee semantics while preserving useful local decisions at the step level across realistic deployment settings, thereby making every certificate interpretable at the same trajectory granularity as the failure event it is intended to control and the downstream action it ultimately informs, under the specified risk target and the benchmark's released annotation semantics across the deployment conditions reported here. This alignment keeps certification interpretable across heterogeneous agent trajectories.

AI USE STATEMENT

Generative AI tools were used to assist with language editing, manuscript organization, and limited programming support during the preparation of this work. In particular, such tools were used to suggest revisions for clarity, identify possible inconsistencies in presentation, and assist with routine code drafting and debugging. They were not used to generate benchmark annotations, fabricate experimental observations, determine reported numerical results, or replace the authors' evaluation of the scientific claims. All AI-assisted text and code were reviewed by the authors. Experimental outputs were obtained from the implemented pipelines and checked against the reported tables and analyses, and the authors take responsibility for the final manuscript, code, results, and conclusions.

ETHICS STATEMENT

This work studies reliability mechanisms for process verification in language model agents. The experiments use existing research benchmarks and their released annotations; we do not conduct new studies involving human subjects or collect personally identifiable information. CPV controls risk relative to the error labels supplied by these benchmarks, so its guarantees should not be interpreted as guarantees of safety beyond the annotated failure definition. In particular, systematic annotation errors, unobserved failure modes, or deployment distributions that differ substantially from the calibration distribution can weaken the practical interpretation of a certified decision.

The method is intended to support safer allocation of automated or human review rather than to eliminate oversight in high-stakes applications. A calibrated verifier may still fail when its assumptions are violated, and certification should therefore not be treated as an unconditional assertion that an agent action is harmless or factually correct. More generally, process verification technology can be used both to improve reliability and to increase the autonomy of deployed systems. We therefore recommend that risk targets, error definitions, and downstream intervention policies be selected in accordance with the consequences of failure in the intended application.

REPRODUCIBILITY STATEMENT

We provide the implementation of CPV, including trajectory-level loss construction, Learn-then-Test calibration, conformal risk control, frozen representation probing, evaluation metrics, and the experimental procedures used in the paper. Section 4.1 describes the datasets, data splits, process scorers, calibration baselines, and evaluation metrics. The calibration objectives and assumptions are defined in Sections 3.2–3.4, and the finite-sample guarantees are stated in Section 3.4, with additional derivations in Appendix F. The supplementary material records dataset-specific preprocessing, hyperparameters, seeded-split protocols, additional risk-control experiments, distribution-shift experiments, and online adaptation. Fixed random seeds and deterministic data-splitting procedures are used where applicable. The released experimental code and result files are organized so that the principal tables and reported summary statistics can be regenerated from the corresponding experiment outputs for independent verification by other researchers.

REFERENCES

AI@Meta. Llama 3 model card, 2024. URL https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

Anastasios N. Angelopoulos, Stephen Bates, Adam Fisch, Lihua Lei, and Tal Schuster. Conformal risk control. In *The Twelfth International Conference on Learning Representations*, pp. 55198–55218, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/f3549ef9b5ff520a7e41ff3cc306ab2b-Abstract-Conference.html.

Anastasios N. Angelopoulos, Stephen Bates, Emmanuel J. Candès, Michael I. Jordan, and Lihua Lei. Learn then test: Calibrating predictive algorithms to achieve risk control. *The Annals of Applied Statistics*, 19(2):1641–1662, 2025. doi: 10.1214/24-AOAS1998. URL https://doi.org/10.1214/24-AOAS1998.

Amos Azaria and Tom Mitchell. The internal state of an LLM knows when it's lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 967–976, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.68. URL https://aclanthology.org/2023.findings-emnlp.68/.

Sher Badshah, Ali Emami, and Hassan Sajjad. SCOPE: Selective conformal optimized pairwise LLM judging. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.13110.

Stephen Bates, Anastasios N. Angelopoulos, Lihua Lei, Jitendra Malik, and Michael I. Jordan. Distribution-free, risk-controlling prediction sets. *Journal of the ACM*, 68(6):43:1–43:34, 2021. doi: 10.1145/3478535. URL https://doi.org/10.1145/3478535.

Hyungjoo Chae, Seonghwan Kim, Junhee Cho, Seungone Kim, Seungjun Moon, Gyeom Hwangbo, Dongha Lim, Minjin Kim, Yeonjun Hwang, Minju Gwak, Dongwook Choi, Minseok Kang, Gwanhoon Im, ByeongUng Cho, Hyojun Kim, Jun Han, Taeyoon Kwon, Minju Kim, Beong-woo Kwak, Dongjin Kang, and Jinyoung Yeo. Web-Shepherd: Advancing PRMs for reinforcing web agents. In *Advances in Neural Information Processing Systems*, volume 38, pp. 63314–63356. Curran Associates, Inc., 2025. doi: 10.52202/085713-2127. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/5bac36213824c6e0b0a173df8e2276a6-Abstract-Conference.html.

Matt Y. Cheung, Ashok Veeraraghavan, Hanjie Chen, and Guha Balakrishnan. Conformal certification of reasoning trace prefixes. *arXiv preprint arXiv:2605.30085*, 2026. URL https://arxiv.org/abs/2605.30085.

Shengda Fan, Xuyan Ye, Yupeng Huo, Zhi-Yuan Chen, Yiju Guo, Shenzhi Yang, Wenkai Yang, Shuqi Ye, Jingwen Chen, Haotian Chen, Xin Cong, and Yankai Lin. AgentProcessBench: Diagnosing step-level process quality in tool-using agents. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pp. 8823–8834. Association for Computing Machinery, 2026. doi: 10.1145/3770855.3817494. URL https://dl.acm.org/doi/10.1145/3770855.3817494.

Isaac Gibbs and Emmanuel J. Candès. Adaptive conformal inference under distribution shift. In *Advances in Neural Information Processing Systems*, volume 34, pp. 1660–1672. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/0d441de75945e5acbc865406fc9a2559-Abstract.html.

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

Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. Process reward models that think. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=FPVCb0WMuN.

Hamed Khosravi and Xiaoming Huo. Conformal selective acting: Anytime-valid risk control for RLVR-trained LLMs. *arXiv preprint arXiv:2605.20270*, 2026. URL https://arxiv.org/abs/2605.20270.

Leslie Kish. *Survey Sampling*. John Wiley & Sons, New York, 1965.

Hadas Orgad, Michael Toker, Zorik Gekhman, Roi Reichart, Idan Szpektor, Hadas Kotek, and Yonatan Belinkov. LLMs know more than they show: On the intrinsic representation of LLM hallucinations. In *The Thirteenth International Conference on Learning Representations*, pp. 66880–66913, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/a712d461e57201efe35d429a6f1731c1-Abstract-Conference.html.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang,

Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

Ryan J. Tibshirani, Rina Foygel Barber, Emmanuel J. Candès, and Aaditya Ramdas. Conformal prediction under covariate shift. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/8fb21ee7a2207526da55a679f0332de2-Abstract.html.

Vladimir Vovk, David Lindsay, Ilia Nouretdinov, and Alex Gammerman. Mondrian confidence machine. Technical Report Working Paper 4, Royal Holloway, University of London, March 2003. URL https://pure.royalholloway.ac.uk/en/publications/mondrian-confidence-machine/.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

Yu Xia, Jingru Fan, Weize Chen, Siyu Yan, Xin Cong, Zhong Zhang, Yaxi Lu, Yankai Lin, Zhiyuan Liu, and Maosong Sun. AgentRM: Enhancing agent generalization with reward modeling. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19277–19290, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.945. URL https://aclanthology.org/2025.acl-long.945/.

Yunpeng Xu, Wenge Guo, and Zhi Wei. Selective conformal risk control. *arXiv preprint arXiv:2512.12844*, 2025. URL https://arxiv.org/abs/2512.12844.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. Which agent causes task failures and when? On automated failure attribution of LLM multi-agent systems. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 76583–76599. PMLR, 2025a. URL https://proceedings.mlr.press/v267/zhang25cq.html.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10495–10516, Vienna, Austria, July 2025b. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.547. URL https://aclanthology.org/2025.findings-acl.547/.

Jian Zhao, Runze Liu, Kaiyan Zhang, Zhimu Zhou, Junqi Gao, Dong Li, Jiafei Lyu, Zhouyi Qian, Biqing Qi, Xiu Li, and Bowen Zhou. GenPRM: Scaling test-time compute of process reward models via generative reasoning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(41):34932–34940, 2026. doi: 10.1609/aaai.v40i41.40797. URL https://ojs.aaai.org/index.php/AAAI/article/view/40797.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. ProcessBench: Identifying process errors in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1009–1024, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.50. URL https://aclanthology.org/2025.acl-long.50/.

A Additional Experimental Details

A.1 Dataset Harmonization

CPV operates on trajectories rather than assuming a benchmark-specific data format. For each dataset, we convert the released annotations into a sequence of steps and preserve the original trajectory identifier so that no trajectory is divided across calibration and evaluation sets. Each step contains the verifier input text, the released annotation or nonjudgeable marker, and metadata required for dataset-specific evaluation and exact dataset reconstruction in all reported experiments.

AgentProcessBench (Fan et al., 2026) provides ternary step labels $y_{it} \in \{-1, 0, +1\}$. For APB, every step with released label -1 is treated as an erroneous step for both the selective false-safe objective and trajectory-contamination evaluation:

$$e_{it} = e_{it}^{\text{caus}} = \mathbf{1}[y_{it} = -1].$$

The resulting trajectory-contamination loss evaluates whether any APB-labeled erroneous step is certified within a trajectory during deployment under the benchmark's released error definition.

For first-error localization, we separately record the earliest -1 step in each APB trajectory as `first_error_idx`. This index defines the FirstErrAcc target used in every evaluation split.

Who&When (Zhang et al., 2025a) directly identifies the decisive step associated with a failed trajectory and is therefore used for both decisive-step localization and trajectory-level evaluation. DeltaBench (He et al., 2025) provides section-level annotations over substantially longer reasoning sequences; each annotated section is treated as one verification step while preserving the full reasoning chain as the trajectory unit.

Missing or nonjudgeable annotations are excluded through the mask m_{it} rather than being assigned a synthetic correct or incorrect label. This choice ensures that each reported risk corresponds only to the annotation semantics provided by the benchmark without introducing inferred labels.

A.2 Trajectory-Aware Baselines and Oracle Diagnostic

All additional APB baselines use the same trajectory-disjoint splits, scorer, risk target, and held-out evaluation protocol as the main experiment. The union-bound baseline allocates a per-step budget α/T_i within trajectory i. The independence-based baseline uses the per-step budget $1 - (1-\alpha)^{1/T_i}$, corresponding to the value that would yield trajectory risk α under independent step events. The max-score aggregation baseline compresses the step scores within each trajectory to a trajectory-level maximum before trajectory-level calibration. CROP (Cheung et al., 2026) provides a structured-prefix trajectory-aware comparator under the same risk target and evaluation protocol.

To decompose certification conservatism, the oracle-score diagnostic replaces the learned process score by an oracle ranking signal derived from the process annotations while leaving the CPV LTT-HB calibration rule, calibration-set size, risk target, and held-out evaluation protocol unchanged. This diagnostic is not deployable and is used only to separate the loss due to finite-sample calibration from the loss due to imperfect scorer ranking in this decomposition.

A.3 Data Splits

For AgentProcessBench and DeltaBench, the data are separated into training, calibration, and test subsets using trajectory-level splits. The default proportions are 0.50, 0.25, and 0.25, respectively. The process scorer is fitted only on the training subset. The calibration subset is then used to select the CPV threshold, after which the threshold remains fixed for evaluation on the held-out test trajectories used for final reporting.

The main AgentProcessBench experiment is repeated over five random split seeds. No step from a trajectory appearing in one partition is allowed to appear in another partition. For experiments with task-family metadata, splitting preserves the intended leakage controls used in the corresponding experiment. Who&When is evaluated over five random seeds because of its smaller sample size; the trajectory-level calibration analysis uses 36 or 37 trajectories in each seeded calibration split. The corresponding seeded partition is reused for the reported decisive-step localization and held-out trajectory-risk analysis in the main paper.

All threshold selection, dimensionality reduction, and probe fitting are performed without access to the held-out test labels during model development, calibration, or evaluation.

A.4 Frozen Representation Probe

For the default lightweight verifier, the current step is concatenated with the task context and a fixed amount of preceding trajectory context. The resulting sequence is passed through a frozen instruction-tuned language model, and the final-layer hidden representation is extracted for the step. Hidden representations are cached before fitting the probe under every reported split.

Principal component analysis is fitted on the training representations only. Unless otherwise specified, the retained dimension is 256. The reduced features are passed to an ℓ_2-regularized logistic classifier using class-balanced training weights. The language-model backbone remains frozen throughout the experiment and every calibration run reported here.

The resulting probability of error is used as the nonconformity score s_{it}. All CPV calibration procedures operate on these cached scalar scores, so changing α, δ, or the calibration procedure does not require rerunning the language model or recomputing cached representations.

A.5 Calibration Grid and Selection

LTT evaluates a finite set of candidate score thresholds $\Lambda = \{\lambda_1, \ldots, \lambda_M\}$. Candidate thresholds are constructed from the calibration scores so that the grid spans the attainable certification decisions. For each threshold, one bounded loss is first computed per trajectory. The empirical risk and corresponding valid p-value are then evaluated across calibration trajectories.

Unless otherwise stated, the main LTT result uses the Hoeffding–Bentkus construction with Bonferroni familywise-error correction. For binary trajectory contamination, we additionally evaluate the exact binomial test. Among all thresholds certified at the requested (α, δ) level, CPV selects the threshold with the largest certified-step rate under this selection rule.

CRC is applied only to monotone trajectory contamination loss. It chooses the largest threshold satisfying the finite-sample corrected empirical-risk criterion in Equation 8 for calibration.

A.6 Evaluation Metrics

For a selected threshold, we report

$$\widehat{R}_{\mathrm{traj}} = \frac{1}{n_{\mathrm{test}}} \sum_{i=1}^{n_{\mathrm{test}}} L_i^{\mathrm{traj}}(\lambda) \tag{9}$$

as empirical trajectory-contamination risk.

Certification efficiency is measured by

$$\widehat{C} = \frac{\sum_i \sum_t m_{it} A_\lambda(x_{it})}{\sum_i \sum_t m_{it}}. \tag{10}$$

We additionally use the fully certified trajectory rate

$$\widehat{C}_{\mathrm{full}} = \frac{1}{n_{\mathrm{test}}} \sum_{i=1}^{n_{\mathrm{test}}} \mathbf{1}[A_\lambda(x_{it}) = 1 \text{ for every judgeable step } t]\,. \tag{11}$$

The two quantities capture complementary deployment utility: $\widehat{C}$ measures the fraction of step-level review decisions avoided, whereas $\widehat{C}_{\mathrm{full}}$ measures the fraction of complete trajectories requiring no flagged-step intervention during deployment at the selected threshold.

Table 7: Summary of the principal evaluation protocols. "Seeded splits" refer to independent trajectory-level repartitioning; Who&When instead uses five seeded evaluation runs because of its smaller sample size. Synthetic draws use fresh generated trajectories rather than a fixed dataset total or a fixed collection of trajectories.

Dataset	Trajectories	Split protocol	Calibration unit	Main target
APB	1,000	$50/25/25$, 5 seeded splits	Trajectory	$\alpha = \delta = 0.1$
Who&When	184	5 seeds	Trajectory	$\alpha = 0.1$
DeltaBench	1,236	$50/25/25$	Trajectory	$\alpha = 0.1$
Synthetic	Not fixed	120 matched draws	Trajectory	$\alpha = 0.1$

For seeded-split experiments, "sat." records the number of held-out test splits whose empirical trajectory risk is no larger than the requested target α. This empirical count is a diagnostic and should not be confused with the formal LTT confidence statement, which concerns population risk over the calibration randomness across repeated draws under this protocol.

Underlying scorer quality is measured using AUROC and AUPRC. For first-error localization, we report the fraction of trajectories for which the highest-risk predicted step coincides with the benchmark-defined first-error target. On APB, this target is the recorded earliest -1 step; on Who&When, it is the annotated decisive failure step.

A.7 Implementation and Reproducibility Summary

Table 7 summarizes the experimental units and calibration protocols used throughout the paper. All data partitioning is performed at the trajectory level whenever a train/calibration/test split is used, and all calibration procedures operate only on the designated calibration trajectories.

For APB and DeltaBench, the default process scorer is trained only on the training partition, threshold selection uses only the calibration partition, and all reported deployment risks are evaluated after the threshold is fixed. The main APB scorer uses frozen Qwen2.5-7B representations, PCA with 256 retained dimensions, and an ℓ_2-regularized class-balanced logistic probe. The same cached scalar scores are reused across the calibration procedures being compared, so differences among LTT, CRC, and the trajectory-aware baselines arise from the calibration rule rather than from rerunning the language model or recomputing its representations.

For Who&When, each of the five seeded runs reuses its corresponding partition for decisive-step localization and trajectory-level evaluation, with 36 or 37 trajectories used in each seeded calibration split. For the synthetic guarantee study, LTT-HB and CRC are evaluated on matched calibration draws with the same score orientation, threshold grid, and large test pool. This matched design isolates the difference in guarantee semantics and threshold selection.

B Additional Risk-Control Results

B.1 Step-Level and Trajectory-Level Calibration

The main paper shows that conventional calibration of pooled steps does not control trajectory contamination. The distinction arises because the two procedures estimate different statistical objects. A step-level procedure acts on observations of the form

$$(x_{it}, e_{it}),$$

whereas CPV constructs one loss observation

$$L_i^{\mathrm{traj}}(\lambda)$$

for every complete trajectory used as the calibration unit.

Even if a threshold achieves a small marginal false-safe probability for a randomly selected step, a trajectory containing several opportunities for a false-safe decision can have a substantially larger probability of at least one accepted error. The discrepancy can grow further when trajectory lengths vary or when errors within a trajectory are dependent.

The repeated AgentProcessBench experiments make this distinction visible. At $\alpha = 0.1$, step-level split conformal calibration obtains mean empirical trajectory contamination 0.169 ± 0.030 and satisfies the test target in $0/5$ seeded splits. Trajectory-level LTT with the exact binomial test obtains

0.054 ± 0.011 risk and satisfies the test target in $5/5$ seeded splits, while LTT with Hoeffding–Bentkus obtains 0.051 ± 0.013 and also satisfies the target in $5/5$ seeded splits under the same test protocol and target definition.

B.2 Risk–Certification Frontier

The requested trajectory risk budget controls the degree of selective certification. Using CRC on AgentProcessBench, target levels $\alpha \in \{0.05, 0.10, 0.20\}$ yield empirical test risks 0.033, 0.091, and 0.177, respectively. The corresponding certification rates are 0.093, 0.219, and 0.306.

This frontier illustrates that the risk target is operational rather than merely diagnostic. A smaller value of α induces a more conservative verification policy, whereas larger tolerated risk permits more steps to pass without additional review during deployment.

B.3 Calibration Sample Size

Finite-sample calibration becomes less conservative as the calibration set grows. In the dedicated AgentProcessBench calibration-size ablation, which is separate from the default $50/25/25$ split, using $n_{\text{cal}} = 75$ trajectories gives a certified-step rate of 0.101 under the default risk target. Increasing the calibration set to $n_{\text{cal}} = 295$ raises certification to 0.151 while preserving empirical target satisfaction across all five reported runs.

This pattern is consistent with the role of finite-sample uncertainty in CPV. Smaller calibration sets produce wider uncertainty about population risk, limiting how aggressively the verifier can certify steps safely under the stated target in every reported run.

B.4 Choice of Finite-Sample Test

We compare simple Hoeffding bounds, exact binomial testing, and the Hoeffding–Bentkus construction. The simple Hoeffding procedure is sufficiently conservative in the principal setting that it certifies no steps. Exact binomial and Hoeffding–Bentkus testing certify 14.9% and 14.6% of steps, respectively, while satisfying the empirical trajectory-risk target in all five runs.

The comparison indicates that the finite-sample test influences usable efficiency but does not alter the central requirement that the loss be constructed at the trajectory level throughout testing.

B.5 Synthetic Guarantee Check

We evaluate both calibration procedures on the same synthetic trajectory-contamination loss. The comparison uses 120 matched independent calibration draws, target $\alpha = 0.1$, a common threshold grid and score orientation, and a common large test pool for evaluating the selected rule. For LTT-HB, we additionally set $\delta = 0.1$ and record whether the selected rule exceeds the population-risk target on the common test pool.

Across the 120 matched draws, LTT-HB has $0/120$ such violations. Its mean test trajectory risk is 0.0360 (standard deviation 0.0145), the maximum is 0.0803, and its mean step-certification rate is 0.0119 (standard deviation 0.0047). CRC has mean test trajectory risk 0.0862 (standard deviation 0.0166), maximum risk 0.1348, and mean step-certification rate 0.0291 (standard deviation 0.0061). Fifteen of the 120 CRC draws exceed 0.1 on test risk; this count is descriptive rather than a violation count because CRC controls expected risk over calibration randomness.

The ordering is internally consistent with the monotonicity of trajectory contamination: on all 120 matched draws, the LTT-HB threshold is no larger than the CRC threshold, and LTT-HB correspondingly has lower trajectory risk and lower certification efficiency. These experiments serve a different purpose from the held-out benchmark results. Benchmark test sets assess empirical deployment behavior on real annotated trajectories, while the synthetic experiment directly checks the finite-sample statistical statements under repeated independent calibration. The synthetic generator is not parameterized to reproduce the APB score distribution, trajectory-length distribution, or attainable certification rate. Its certification efficiency should therefore be interpreted within this controlled guarantee check rather than as an estimate of deployment efficiency on APB.

B.6 Synthetic Protocol and Scope

The synthetic experiment is designed as a repeated-calibration check rather than as a model of the empirical APB distribution. Each of the 120 independent repetitions produces a fresh calibration sample, after which LTT-HB and CRC are applied to the same monotone trajectory-contamination loss. Within each matched repetition, both procedures use the same score orientation, the same finite candidate threshold grid, and the same large test pool for evaluating the population trajectory risk of the selected threshold under the common evaluation protocol.

The comparison is therefore paired at the calibration-draw level. If $\widehat{\lambda}^{(b)}_{\mathrm{LTT}}$ and $\widehat{\lambda}^{(b)}_{\mathrm{CRC}}$ denote the selected thresholds in draw b, then the observed ordering

$$\widehat{\lambda}^{(b)}_{\mathrm{LTT}} \leq \widehat{\lambda}^{(b)}_{\mathrm{CRC}}, \qquad b = 1, \ldots, 120,$$

is consistent with the monotonicity of trajectory contamination and with the more conservative high-probability LTT criterion. The corresponding risk and certification summaries are reported in Section B for a direct quantitative comparison.

The synthetic generator is intentionally not calibrated to match the APB score distribution, trajectory-length distribution, or attainable certification rate. Consequently, the synthetic certification values are not intended to predict APB deployment efficiency. Their role is to test the finite-sample statements under repeated independent calibration while keeping the two procedures on the same loss, threshold grid, score orientation, and evaluation population.

C Distribution Shift and Weighted Calibration

The guarantees in the main paper assume exchangeability between calibration and deployment trajectories. Agent systems can violate this assumption when the task distribution, generator, or trajectory composition changes after calibration. Motivated by weighted conformal calibration under covariate shift (Tibshirani et al., 2019), we study a weighted extension as an empirical robustness mechanism rather than as part of the primary finite-sample guarantee.

Let $w_i \geq 0$ denote the importance weight assigned to calibration trajectory i. The weighted empirical risk is

$$\widehat{R}_w(\lambda) = \frac{\sum_{i=1}^{n} w_i L_i(\lambda)}{\sum_{i=1}^{n} w_i}. \tag{12}$$

The implementation summarizes weight concentration through the Kish effective sample size (Kish, 1965)

$$n_{\mathrm{eff}} = \frac{\left(\sum_i w_i\right)^2}{\sum_i w_i^2}. \tag{13}$$

The weighted procedure substitutes this effective sample size into the finite-sample testing calculation and chooses the most efficient threshold that passes the resulting criterion.

On the main AgentProcessBench seeded-split experiment, weighted LTT obtains mean trajectory risk 0.047 ± 0.013, certifies 14.0% of steps, and satisfies the empirical test target in all five seeded splits under the reported evaluation protocol and fixed risk target.

We emphasize that this weighted construction is a practical finite-sample surrogate. The formal guarantee stated for standard CPV in the main paper applies under exchangeability and should not be transferred automatically to arbitrary covariate or generator shift. The weighted experiment instead measures whether importance weighting can improve robustness using observed distributional information available at calibration time.

D Conditional and Groupwise Calibration

A global risk guarantee can mask differences among task families or deployment domains. We therefore evaluate a Mondrian-style extension (Vovk et al., 2003) that calibrates thresholds separately within predefined groups. Let $G_i \in \mathcal{G}$ identify the group associated with trajectory i. For each group g, CPV constructs the corresponding calibration subset

$$\mathcal{I}_g = \{i : G_i = g\}$$

and performs trajectory-level calibration using only trajectories from that group.

The resulting deployment rule uses

$$A_{\lambda_g}(x_{it})$$

for trajectories belonging to group g. This permits heterogeneous certification thresholds when score distributions or error prevalence differ substantially across domains.

On AgentProcessBench, the Mondrian variant is conservative: it yields mean trajectory contamination 0.008 ± 0.006 and certifies approximately 3.0% of steps while satisfying the empirical target in all five reported seeded splits. The reduction in efficiency reflects the smaller number of calibration trajectories in each group after partitioning the calibration set.

The result illustrates a general tradeoff. More granular guarantees may better reflect heterogeneous deployment populations, but partitioning the calibration sample increases finite-sample uncertainty. The global trajectory-level CPV procedure is therefore our default unless group-specific risk constraints are required by the application under the stated deployment policy.

E Online Adaptation

Static calibration assumes that the deployment distribution remains sufficiently stable after the threshold has been selected. We additionally study an online extension based on adaptive conformal inference (ACI) (Gibbs & Candès, 2021), in which the effective calibration level is updated sequentially in response to observed trajectory losses.

Let α_t denote the internal risk level before processing deployment trajectory t. After observing the realized loss L_t, the level is updated according to the following bounded update:

$$\alpha_{t+1} = \Pi_{[\alpha_{\min}, \alpha_{\max}]} \left[\alpha_t + \gamma \left(\alpha - L_t \right) \right], \tag{14}$$

where $\gamma > 0$ is a learning rate and Π denotes projection onto a permitted interval. The current value α_t determines the threshold used for the next trajectory without refitting the scorer.

The purpose of this experiment is not to replace the static finite-sample statement in the main paper. Rather, it examines whether trajectory-level risk feedback can support adaptation under sequential deployment. In the reported online experiment, the long-run empirical trajectory risk converges to 0.099 for a target $\alpha = 0.1$.

Online adaptation introduces additional assumptions and design choices, including the observation delay for labels, the update rate, and the temporal structure of the deployment sequence. For this reason, we treat it as an extension rather than part of the primary CPV guarantee.

F Proofs

F.1 Finite-Sample LTT p-Values

For a candidate threshold λ, define its empirical risk by the following sample average:

$$\widehat{R}_n(\lambda) = \frac{1}{n} \sum_{i=1}^{n} L_i(\lambda), \qquad 0 \le L_i(\lambda) \le 1,$$

and consider the corresponding null hypothesis

$$H_\lambda : R(\lambda) > \alpha.$$

For bounded trajectory losses, the Hoeffding–Bentkus construction used in the main experiments is (Angelopoulos et al., 2025)

$$p_{\mathrm{HB}}(\lambda) = \min \left\{ \exp\Big[-n\, h_1\Big(\widehat{R}_n(\lambda) \wedge \alpha, \alpha \Big) \Big],\ e\, \mathbb{P}\Big[\mathrm{Bin}(n, \alpha) \le \Big\lceil n \widehat{R}_n(\lambda) \Big\rceil \Big] \right\}, \tag{15}$$

where

$$h_1(a, b) = a \log \frac{a}{b} + (1 - a) \log \frac{1 - a}{1 - b},$$

with the usual continuous extension at $a \in \{0, 1\}$ and $a \wedge b = \min\{a, b\}$.

For binary trajectory-contamination losses, let

$$K_\lambda = \sum_{i=1}^{n} L_i^{\mathrm{traj}}(\lambda).$$

The exact one-sided binomial p-value is then defined by the following finite sum:

$$p_{\mathrm{bin}}(\lambda) = \mathbb{P}[\mathrm{Bin}(n, \alpha) \le K_\lambda] = \sum_{k=0}^{K_\lambda} \binom{n}{k} \alpha^k (1-\alpha)^{n-k}. \tag{16}$$

For a finite candidate set Λ, the main LTT procedure applies Bonferroni familywise-error control and rejects H_λ whenever

$$p_\lambda \le \frac{\delta}{|\Lambda|}.$$

Among the rejected thresholds, CPV selects the threshold with the largest certified-step rate.

F.2 LTT and CRC Guarantee Semantics

LTT and CRC control the same trajectory-contamination loss under different finite-sample statements. For LTT, calibration randomness is explicit in the high-probability guarantee:

$$\mathbb{P}\left(R_{\mathrm{traj}}(\widehat{\lambda}_{\mathrm{LTT}}) \le \alpha\right) \ge 1 - \delta.$$

Thus, under the stated assumptions, the probability that calibration selects a threshold whose population trajectory risk exceeds α is at most δ over calibration draws.

For CRC with monotone trajectory contamination, the corresponding guarantee is instead

$$\mathbb{E}\left[L_{\mathrm{test}}^{\mathrm{traj}}(\widehat{\lambda}_{\mathrm{CRC}})\right] \le \alpha.$$

The expectation averages over the calibration sample and a future exchangeable test trajectory. This statement does not require the population risk associated with every realized calibration draw to lie below α. Accordingly, the number of individual synthetic draws with evaluated risk above α is a violation count for the high-probability LTT statement, but only a descriptive diagnostic for CRC.

These semantics also explain the empirical ordering in the matched synthetic study. LTT-HB selects a threshold no larger than the CRC threshold in all 120 draws, producing lower mean trajectory risk and lower mean certification. The difference reflects the strength and form of the guarantee rather than a change in the underlying score or loss.

F.3 Proof of Theorem 1

For each candidate threshold $\lambda \in \Lambda$, define the null hypothesis

$$H_\lambda : R(\lambda) > \alpha.$$

By assumption, the p-value associated with each H_λ is valid under the null. Let $\widehat{\Lambda}$ denote the set of thresholds whose null hypotheses are rejected after applying a familywise-error-controlling multiple testing procedure at level δ.

Familywise error control then provides the following bound:

$$\mathbb{P}\left(\exists \lambda \in \widehat{\Lambda} \text{ such that } R(\lambda) > \alpha\right) \le \delta. \tag{17}$$

Consequently, with probability at least $1 - \delta$, every threshold in $\widehat{\Lambda}$ satisfies

$$R(\lambda) \le \alpha.$$

The selected efficiency-maximizing threshold is

$$\lambda^\star = \arg\max_{\lambda \in \widehat{\Lambda}} \mathrm{Eff}(\lambda)$$

belongs to $\widehat{\Lambda}$ whenever the controlling set is nonempty. Therefore, on the event that no false rejection occurs,

$$R(\lambda^\star) \le \alpha.$$

Combining this observation with Equation 17 yields the bound

$$\mathbb{P}\left(R(\lambda^\star) \le \alpha\right) \ge 1 - \delta.$$

This proves Theorem 1 under the stated exchangeability assumptions.

The argument depends only on the validity of the trajectory-level p-values and familywise error control. It does not require independence among steps within a trajectory because the testing procedure receives the bounded trajectory losses

$$L_1(\lambda), \ldots, L_n(\lambda)$$

as its observations for calibration in the familywise test used here.

F.4 Monotonicity of Trajectory Contamination

For completeness, consider any thresholds $\lambda_1 \le \lambda_2$. From Equation 3,

$$A_{\lambda_1}(x_{it}) \le A_{\lambda_2}(x_{it})$$

for every step because increasing the threshold can only enlarge the set of certified steps.

Suppose first that

$$L_i^{\mathrm{traj}}(\lambda_1) = 1.$$

Then there exists some step t for which

$$e_{it}^{\mathrm{caus}} = 1 \qquad \text{and} \qquad A_{\lambda_1}(x_{it}) = 1.$$

Since $A_{\lambda_2}(x_{it}) \ge A_{\lambda_1}(x_{it})$, the same step is also certified at λ_2. Hence

$$L_i^{\mathrm{traj}}(\lambda_2) = 1.$$

Therefore, the two trajectory losses obey

$$L_i^{\mathrm{traj}}(\lambda_1) \le L_i^{\mathrm{traj}}(\lambda_2),$$

so trajectory contamination is nondecreasing in the certification threshold. This monotonicity permits the CRC construction used in Section 3.4.

F.5 Why Step-Level Control Does Not Imply Trajectory-Level Control

Consider a trajectory containing T steps. For illustration, suppose that each step has marginal false-safe probability q and that these events are independent. The probability that the trajectory contains at least one false-safe decision is then

$$1 - (1-q)^T. \tag{18}$$

For every $T > 1$ and $q \in (0, 1)$,

$$1 - (1-q)^T > q.$$

Thus, even in this simple independent setting, controlling a per-step error probability at level q does not control trajectory contamination at the same level.

Independence is not assumed by CPV and is unnecessary for the argument. Without independence, the trajectory event remains the following union of stepwise failure events:

$$\mathbf{1}\left[\bigcup_{t=1}^{T_i} \{e_{it}^{\mathrm{caus}} = 1, A_\lambda(x_{it}) = 1\}\right],$$

whose probability is generally not determined by the marginal risk of an individual step. CPV avoids this mismatch by calibrating the union event directly through one loss observation per complete trajectory during calibration and evaluation under the stated protocol throughout.