LEARNING WHEN TO EDIT: IMPROVING PROCESS VERIFICATION THROUGH CALIBRATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Process reward models (PRMs) assess intermediate reasoning steps but may miss errors or reject valid steps. Existing PRMs and generative critics offer alternative judgments, but deciding which alternatives improve a verifier's current predictions remains challenging. We study calibrated step-label editing, which learns to select corrections while keeping the reasoning text and language-model unchanged. Proposed error locations are converted into candidate label edits, and a lightweight selector trained on annotated reasoning traces determines which edits to apply. Experiments on PRMBench and an adapted DeltaBench task show gains across multiple PRMs. Comparisons of candidate sources show that PRM scores provide useful corrections, while the additional value of critics varies across models. On PRMBench, learning an edit's effect on the verification metric yields greater gains than predicting changes in first-error correctness; finer first-error categories offer little additional benefit. These findings support improving process verification through learned selection of existing model judgments.

1 INTRODUCTION

Process reward models (PRMs) score reasoning steps to support solution selection and search (Lightman et al., 2024; Wang et al., 2024). However, their judgments may miss reasoning errors or reject valid steps (Song et al., 2025). Judgments from other PRMs or generative language models can provide candidate corrections, but these alternatives are not necessarily better than the verifier's original predictions. Annotated reasoning traces can be used to train an edit selector that distinguishes useful edits from harmful ones. The practical challenge is to learn which candidate edits improve the verifier's current judgments.

We study *calibrated step-label editing*. The verifier to be improved, called the *Anchor*, supplies an initial vector of step judgments. Candidate corrections are derived from the Anchor's own scores, a reference PRM, or a language-model *Critic*. A common editor converts these candidates into alternative step-label vectors (Figure 1). Calibration fits a selector and sets its acceptance threshold, determining whether to adopt a candidate edit or retain the Anchor's original judgments. This process edits step labels rather than reasoning text, while keeping all language-model fixed.

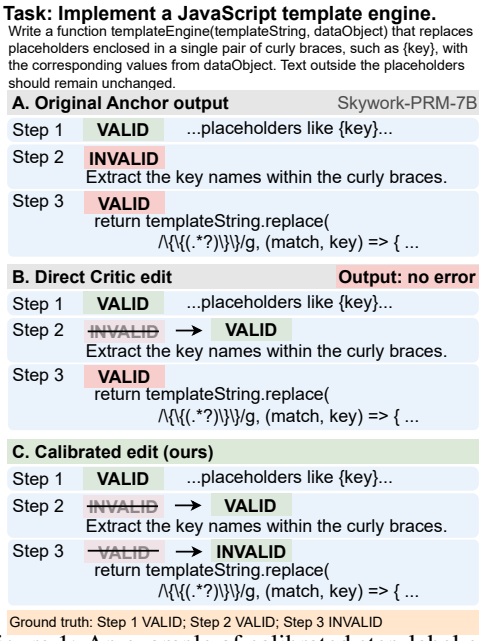


Figure 1: An example of calibrated step-label editing. (A) The Anchor rejects a valid step and misses an error. (B) Direct Critic editing removes only the false alarm. (C) Calibrated editing corrects both judgments.

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.

We examine two design choices within this framework. First, we study *candidate construction* by comparing Anchor-only, reference-only, direct-Critic, and combined candidate pools under the same editor and annotation budget. These comparisons distinguish gains from calibrating existing PRM judgments from the additional value of Critic proposals.

The second choice is the *selection target*. A first-error proposal can change several step judgments. Whether the proposed index is correct therefore does not fully describe the value of adopting it: two proposals that both miss the first error can improve and degrade the edited vector respectively. We compare two-head paired logistic and four-event PAIR-EF, which predict changes in first-error correctness, with native-utility regression (metric-linear), which predicts an edit's marginal effect on the step-level evaluation score. This comparison distinguishes finer modeling of first-error events from learning the consequences of the complete edit.

Post-hoc reward calibration already improves model outputs without retraining the reward model (Huang et al., 2025), and process-advantage verifiers learn relative progress toward a correct solution (Setlur et al., 2025). Work on reflective reasoning also motivates supervision beyond the first error (Yang et al., 2025). Our focus is improving an existing verifier's predictions by selecting *label edits*. Building on learning to defer and metric-aware prediction (Mozannar & Sontag, 2020; Narasimhan et al., 2015), we study this decision through matched candidate and selector comparisons, together with analysis of the resulting changes to step judgments.

Our study makes three contributions:

- **A framework for improving existing process verifiers.** Calibrated step-label editing combines existing model judgments with supervised edit selection. With metric-linear selection, all 15 Anchor Critic pair means improve on PRMBench, averaging +3.98 PRMScore points. It also improves verification performance on an adapted DeltaBench task.
- **Evidence on useful candidate sources.** We compare candidate sources using the same editor, selector, and annotation budget. Reference-PRM candidates improve verification performance on PRMBench. Combining direct and reranked Critic proposals yields similar average gains. Using Qwen2.5 Critic proposals yields a 1.17-point PRMScore gain over using candidates derived from reference-PRM scores.
- **An explanation of selection-target differences.** Experiments show that learning step-level edit utility yields greater gains than refining first-error event categories. On PRMBench, metric-linear improves over paired logistic by 0.49 PRMScore points with the same combined candidate pool. Four-event selection performs similarly to paired logistic.

2 Related Work

Process supervision and verification targets. Process supervision trains verifiers using human step annotations (Lightman et al., 2024) or automatically constructed feedback (Wang et al., 2024). Generative verifiers learn to reason during assessment (Zhang et al., 2025a; She et al., 2025; Zhao et al., 2026), while ReasonEval and PathFinder-PRM distinguish validity, redundancy, and semantic error types (Xia et al., 2025; Pala et al., 2025). Existing benchmarks assess different aspects of process verification. ProcessBench focuses on earliest-error localization, PRMBench evaluates fine-grained step judgments, and DeltaBench assesses error detection in long reasoning traces across domains (Zheng et al., 2025; Song et al., 2025; He et al., 2025). Zhang et al. (2025b) analyze differences between response-level and step-level evaluation. Beyond the First Error further introduces error propagation and cessation annotations to train PRMs for self-correcting trajectories (Yang et al., 2025). We instead examine whether first-error correctness provides sufficient supervision for selecting step-label edits. First-error correctness alone does not fully capture an edit's effect on overall verification quality, because edits with the same first-error outcome can produce different step-label predictions.

Critic feedback and complementary model selection. Studies of intrinsic self-correction and tool-assisted CRITIC highlight the importance of feedback for effective correction (Huang et al., 2024; Gou et al., 2024). Learning-to-defer methods learn when to delegate decisions to an alternative decision maker (Madras et al., 2018; Mozannar & Sontag, 2020). RouteLLM learns preference-based language-model routing (Ong et al., 2025), while BayesianRouter selects reward models during alignment (Wu & Lu, 2026). Our work similarly learns when to rely on alternative model judgments.

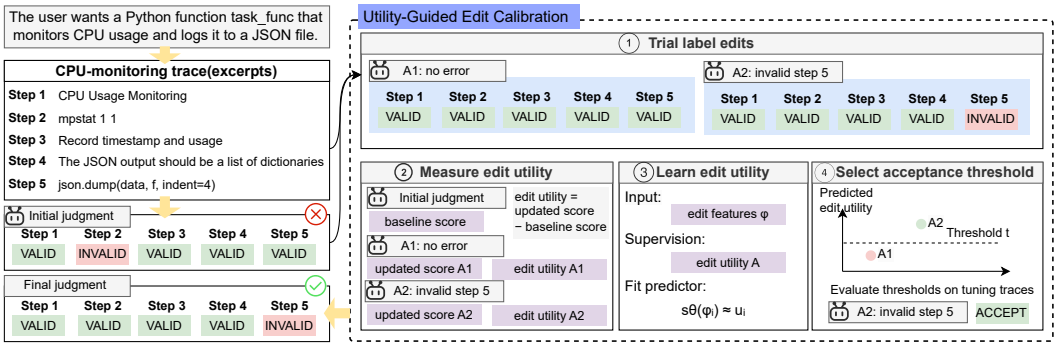


Figure 2: Calibrated step-label editing. (1) Construct candidate edits from Critic proposals. (2–3) Measure their native-score gains on labeled fit traces and learn an edit-utility predictor. (4) Select t on separate tuning traces; at inference, accept the highest-scoring candidate if its predicted utility reaches t, otherwise retain the Anchor labels.

Rather than routing a query or revising an answer, we learn whether a candidate label edit improves the verifier's existing step judgments.

Post-hoc calibration and relative utility. Probability calibration adjusts confidence estimates (Guo et al., 2017); post-hoc reward calibration corrects scalar reward biases without retraining the reward model (Huang et al., 2025). For PRMs, uncertainty calibration estimates continuation success to allocate inference computation (Park et al., 2025). Relative utility also has precedents: Rewarding Progress learns changes in eventual success under a prover policy (Setlur et al., 2025), while the Prefix Utility Model uses solve-rate gains to learn prefix utility through pairwise ranking (Zhou et al., 2026). Both approaches evaluate future reasoning success. Our utility instead measures the effect of editing a verifier's step labels relative to leaving them unchanged. Because the evaluation metric depends on aggregate confusion counts, this objective connects to plug-in prediction and nondecomposable-metric optimization (Narasimhan et al., 2014; 2015). We examine whether learning an edit's contribution to overall verification performance leads to better selection than supervision based only on first-error outcomes.

3 Calibrated Step-Label Editing

The goal is to improve an existing verifier's step judgments using annotated traces and candidate corrections, while keeping language-model weights fixed. The framework separates candidate construction, a deterministic label editor, and learned edit selection. An accepted edit changes the predicted labels of a fixed reasoning trace; otherwise the original prediction is retained. The selectors below share the editor and calibration labels. They differ in whether they predict changes in first-error correctness or the edit's effect on the native verification score. Figure 2 illustrates the calibration and application stages for native-utility selection. Appendix G presents the corresponding trace excerpts, candidate actions, and stepwise outcomes.

3.1 Candidate Actions and Step-Label Editing

For a trace x with L steps, a frozen Anchor emits a native prediction $v_0(x)$ containing validity labels and, when supported, redundancy labels. Its action is the first invalid validity label, $a_0(x) = \min\{j : v_{0,j}^{\text{valid}} = 0\}$, or -1 when every step is valid. In the combined candidate pool, a Critic provides a direct first-error proposal and a reranked proposal from a cached top-three candidate prompt. Equal proposals become one agreement action. We remove actions equal to a_0 and actions outside any emitted label vector. The remaining set $\mathcal{A}(x)$ has at most two actions.

The editor $E(v_0, a)$ maps each proposal to a native vector. For $a = k \geq 0$, it makes the prefix $j < k$ valid, marks k invalid, and preserves the suffix $j > k$. For $a = -1$, all steps become valid. The same semantic change is applied to redundancy labels with their opposite polarity. This operation uses the

predicted index and emitted channels; the benchmark category is consulted only by the evaluator. Rejected actions and unavailable cache entries retain v_0.

3.2 Selection targets from shared calibration labels

Let y be a calibration trace's first-error label and define

$$d(x, a) = \mathbf{1}[a = y] - \mathbf{1}[a_0(x) = y] \in \{-1, 0, 1\}. \tag{1}$$

A fix has $d = 1$, a break has $d = -1$, and $d = 0$ indicates unchanged first-error correctness.

Paired logistic. Two binary logistic models estimate $p_+(x, a) = \Pr(d = 1 \mid \phi)$ and $p_-(x, a) = \Pr(d = -1 \mid \phi)$. Their difference $s_{\mathrm{PL}} = p_+ - p_-$ ranks proposals by predicted relative correctness. For actions disagreeing with a_0, proposal correctness implies a fix and Anchor correctness implies a break. Therefore correctness-difference logistic deferral has exactly these two training targets under the same feature and fitting choices.

Four-event selection (PAIR-EF). With $c = \mathbf{1}[y = -1]$, four binary logistic heads estimate $p_{c,+} = \Pr(c, d = 1 \mid \phi)$ and $p_{c,-} = \Pr(c, d = -1 \mid \phi)$ for $c \in \{0, 1\}$. The ranking score is

$$s_{\mathrm{EF}} = p_{1,+} - p_{1,-} + p_{0,+} - p_{0,-}. \tag{2}$$

The class c defines training targets and is absent from inference inputs. The four heads are fitted independently. Comparing the resulting selector with paired logistic tests whether event subdivision improves edit selection.

At the population level, summing the four true event probabilities yields $\sum_c \{\Pr(c, d = 1 \mid \phi) - \Pr(c, d = -1 \mid \phi)\} = \mathbb{E}[d \mid \phi]$, the same target as paired logistic. Event subdivision changes the parameterization and finite-sample estimation, rather than the underlying prediction target.

Direct native-utility regression. In the official confusion-count convention, native PRMScore is

$$M(C) = \frac{1}{2}\left[\frac{2TP}{2TP + FP + FN} + \frac{2TN}{2TN + FP + FN}\right], \tag{3}$$

with zero contribution for a zero denominator. Let C_{fit} be the aggregate fit confusion count and $c_0(x), c_a(x)$ the record counts before and after an edit. Metric-linear fits a ridge regression to

$$u(x, a) = M(C_{\mathrm{fit}} - c_0(x) + c_a(x)) - M(C_{\mathrm{fit}}). \tag{4}$$

We compute utility targets and fit the regressor using only the fit partition. Using the same annotated traces as the event-based selectors, the target u measures each edit's marginal effect on verification performance, including effects from edits with $d = 0$. The predicted utility $\hat{u}(x, a)$ is used to rank candidate edits. Since M is nonadditive, the acceptance threshold is selected by evaluating the joint effect of the accepted edits on the tuning partition.

3.3 Shared features and selection rule

Let r_j be the empirical quantile of the Anchor's raw validity score at step j, using the sorted validity scores from fit traces as the reference. Define $q(k) = r_k$ for $k \geq 0$ and $q(-1) = \min_j r_j$. All three selectors use

$$\begin{aligned}\phi(x, a) = (1, \mathbf{1}_{\mathrm{gen}}, \mathbf{1}_{\mathrm{rerank}}, \mathbf{1}_{\mathrm{agreement}}, \mathbf{1}[a_0 = -1], \mathbf{1}[a = -1],\\ (a_0 + 1)/L, (a + 1)/L, \mathrm{sgn}(a - a_0), q(a_0), q(a), \min_j r_j, L^{-1} \textstyle\sum_j r_j, \log(1 + L)).\end{aligned} \tag{5}$$

These are trace-wide minimum and mean ranks. Question identifiers and reference error categories are excluded. Logistic heads use regularized binary cross-entropy, and metric-linear uses ridge regression.

For a ranking score s, choose $a^*(x) = \arg\max_{a \in \mathcal{A}(x)} s(x, a)$. A threshold accepts this action when $s(x, a^*) \geq t$; otherwise the Anchor is retained. The threshold maximizes native PRMScore on tuning data. The grouped calibration procedure and all numerical choices are specified together in Section 4.2.

4 EXPERIMENTAL DESIGN

4.1 MODELS, RECORDS, AND CALIBRATION UNITS

We evaluate five PRM Anchors whose step judgments are edited: Qwen2.5-Math-PRM-7B, PathFinder-PRM-7B, ReasonEval-7B, Skywork-PRM-7B, and RLHFlow/Llama3.1-8B-PRM-Mistral-Data. Three generative Critics supply candidate corrections: DeepSeek-Math-7B-Instruct, Llama-3.1-8B-Instruct, and Qwen2.5-7B-Instruct. We evaluate each Anchor with each Critic, giving 15 Anchor Critic pairs. For reranking, we shortlist suspicious steps using scores from a shared reference PRM. These scores are shared across Anchors and were obtained through a separate data-conversion pipeline. The reference scores guide candidate construction, while editing and evaluation operate on each Anchor's own step-label predictions.

We use a fixed version of the native PRMBench reader, which yields 6,969 records: 6,211 modified traces used to compute the official aggregate PRMScore and 758 original correct traces evaluated separately as controls. Among the modified traces, 160 have no annotated erroneous steps. For each Critic, the direct and reranked proposal caches each match 6,192 modified traces. All 758 original correct controls and the remaining 19 modified traces lack compatible cached proposals. Records without compatible proposals retain the Anchor's original predictions without editing.

One calibration unit is one annotated trace with its reference error step set, sufficient for both first error and native step metric targets. It is neither a single binary label nor an independent source question. Multiple traces can share one source problem; the total collection has 1,284 source question components.

4.2 GROUPED CALIBRATION AND FITTING

We group records by shared source IDs or identical questions after whitespace normalization, keeping connected records together. For each seed 0–9, a deterministic label-free ordering allocates about 1,000 records to calibration, split about 50%/25%/25% into fit/tuning/risk partitions. Groups remain indivisible. The remaining 5,964–5,972 records form the target, including original correct controls that do not enter the official aggregate PRMScore. Every model pair uses the same partitions for a given seed.

Logistic heads minimize summed binary cross-entropy with ridge coefficient 5; metric-linear uses ridge 1. Intercepts are unpenalized. Threshold search considers the unique values of 61 tuning-score quantiles and complete abstention. Except for complete abstention, candidate thresholds must select at least five edits on the tuning partition; ties in tuning PRMScore favor fewer interventions. Each split refits the selector and fixes its threshold before target evaluation. Optional variants apply a group-adjusted lower confidence bound (GLCB) check to the selected policy on the separate risk partition. Optimizer, rare-event, and tie-handling details are collected in Appendix A.

4.3 MATCHED CONTROLS AND DIAGNOSTIC COMPARISONS

The Anchor-only control uses its lowest-score step and a no-error action as candidates, with the same native editor, annotation budget, and learned selectors. Additional controls retune a global Anchor score threshold, apply argmin or no-error edits unconditionally, or use tuning data to select a single strategy from Anchor, direct, and rerank for all target records. Confidence gating ranks error proposals by $1 - q(a)$ and no-error proposals by $\min_j r_j$. Direct, rerank, agreement, and no-threshold selection share the same proposal cache. Each method is evaluated on the step-label predictions obtained by applying its selected edits to the Anchor's output.

We compare reference-only, direct-only, and combined candidate pools using the same selectors, data partitions, and editor. The reference-only pool contains the lowest-scoring reference step and a no-error action, while the direct-only pool uses direct Critic proposals. All Critics use the same fixed rerank routing rule: a minimum reference score below 0.5 supplies a reranked Critic proposal; otherwise the rerank branch supplies a no-error candidate. Direct proposals remain available. These comparisons assess candidate sources under shared reranking coverage.

All learned-selector ablations share the candidate pool and calibration partitions. To isolate supervision from learner choice, matched ridge regressors predict first-error utility d, the change in correctly

labeled steps $\Delta(TP + TN)$, or native utility u. They share features, ridge coefficient 1, candidate rules, and threshold tuning. The event audit applies first-error fix, break, and $d = 0$ edit sets separately to the same Anchor and recomputes each set's native effect. The annotation-budget comparison uses 100, 200, 400, 750, and 1,000 traces, including all fit/tuning/risk allocations. Within each seed, every budget is scored on the intersection of its target groups with those of all other budgets. The second-cache replication replaces the DeepSeek and Llama proposals with independently generated outputs, retaining five Anchors, ten splits, and the same 1,000-trace protocol. Its selectors are recalibrated on the replacement proposals.

4.4 Metrics and statistical analysis

We recompute native target-only PRMScore from the official confusion counts. The sufficient-statistic implementation matches saved official scores, and first-split edited predictions are checked with the official evaluator. We report PRMScore and gains over the matched Anchor or comparison policy. Intervention coverage is the fraction of target traces edited; fixes and breaks measure changes in exact first-error correctness. Auxiliary risk comparisons use the informative-event break ratio $B/(F + B)$, defined in Appendix C.1.

Point estimates average the 15 model pairs and ten splits equally. Paired question-group bootstrap intervals and resampling details are reported in Appendix B.

Coverage sweeps inspect fixed rankers on target data and serve as descriptive diagnostics. Deployment tables use tuning-selected thresholds and, where applicable, the subsequent risk-partition GLCB decision.

4.5 Evaluation on DeltaBench

On DeltaBench, the primary comparison evaluates combined candidates against reference-only candidates using metric-linear selection. The combined pool uses Qwen2.5-7B-Instruct as the Critic. Results are averaged over the Qwen2.5-Math-PRM-7B, Skywork-PRM-7B, and RLHFlow Anchors and ten calibration splits. Features, regularization, prompts, candidate rules, the common 0.5 routing gate, threshold search, and analysis are fixed before target evaluation. Selector coefficients and acceptance thresholds are fitted on local calibration data.

The public release contains 1,236 traces. Exact normalized-question overlap with prior local data excludes 137; a label-free full-input context check retains 468 traces in 465 question groups. All native, reference, and Critic inputs must fit their model limits without truncation, including the token limit of the Qwen and Skywork PRMs. Grouped allocation assigns 117 annotated traces to calibration and 351 traces (349 groups; 3,259 sections) to a fixed target. Ten 50/25/25 fit/tuning/risk splits use this same calibration pool. The target covers mathematics, coding, science, and general reasoning.

We use the supplied sections and reasoning-correctness labels. Their pooled binary class-macro-F1 has the form of Equation 3. This is an adapted DeltaBench task: the official evaluator also includes usefulness errors and uses a different averaging convention. All target traces, including all-correct traces, receive model predictions. Every policy's decisions are saved before target scoring. Appendix F provides detailed analyses of candidate sources, domain-level performance, and accepted-edit outcomes.

5 Results

5.1 Candidate Sources and Verification Gains

Table 1 shows gains from calibrated editing across all four candidate pools. Anchor-only selection demonstrates that the verifier's own scores can support useful corrections. Reference-only candidates provide further gains and perform well under all three selectors. With metric-linear selection, the combined pool improves the mean Anchor PRMScore by 3.98 points, achieving performance close to reference-only selection.

The additional value of Critic proposals varies across models. With metric-linear selection, combined Qwen2.5 candidates improve on reference-only selection by 1.17 PRMScore points, while DeepSeek-

Table 1: Candidate source × selector on native PRMBench. Entries are PRMScore, with gain over the original Anchor (62.49) in parentheses. Rows share the editor and calibration budget; columns vary the selection rule.

Candidate pool	PAIR-EF	Paired logistic	Metric-linear
Evaluated Anchor only	64.52 (+2.02)	64.24 (+1.74)	64.46 (+1.97)
Direct Critic candidates	64.83 (+2.34)	64.82 (+2.33)	65.13 (+2.64)
Reference PRM only	66.42 (+3.93)	66.45 (+3.95)	66.57 (+4.08)
Critic direct + rerank	65.97 (+3.47)	65.98 (+3.49)	66.47 (+3.98)

Math and Llama-3.1 yield differences of -0.86 and -0.60 points, respectively. These differences explain why the Qwen-specific gain is not reflected in the pooled comparison between combined and reference-only candidates.

5.2 Selection Objectives and Native-Metric Performance

On PRMBench, native-utility selection provides greater gains than finer first-error event modeling in the combined pool (Table 1). Metric-linear exceeds paired logistic by 0.49 PRMScore points, while PAIR-EF and paired logistic perform similarly. Metric-linear's advantage over paired logistic is smaller with reference-only candidates, while PAIR-EF slightly outperforms metric-linear in the Anchor-only pool.

With the same ridge regressor, native utility yields the highest mean PRMScore among the three targets on PRMBench. It exceeds first-error utility by 0.21 points and correct-step-count change by 0.87 points (Table 2). These results support learning an edit's effect on the native metric as the selection target.

Table 2: Selection-target comparison with combined candidates. All targets share the same ridge regressor, features, regularization, and tuning rule.

Regression target	PRMBench	DeltaBench
First-error utility d	66.26	56.51
Correct-step change $\Delta(TP+TN)$	65.60	54.53
Native utility u	66.47	56.66
Contrast	**PRMBench difference**	**DeltaBench difference**
$u-d$	+0.21	+0.16
$u-\Delta(TP+TN)$	+0.87	+2.13

Selecting when to apply an edit is also important. Unconditionally applying direct Critic proposals scores 61.09, below the original Anchor, while rerank editing reaches 63.74. In the combined pool, accepting every top-ranked PAIR-EF proposal gives -0.16 points, compared with +3.47 after threshold tuning. These comparisons show the benefit of selecting candidate edits before applying them to the Anchor's predictions. The acceptance and event-head ablations are given in Table 9.

Step-level effects of first-error events. For a proposal disagreeing with the Anchor, $d = 0$ means that both first-error indices are incorrect. Applying the accepted $d = 0$ edits alone nevertheless improves PRMScore by 0.83 points for PAIR-EF, 0.84 for paired logistic, and 1.12 for metric-linear (Table 3). Edits can therefore improve step judgments even when they do not recover the first-error index. Metric-linear also accepts more first-error breaks than paired logistic, yet its break set has a smaller score reduction.

These observed edits explain the distinction between the selection targets: a first-error fix, break, or unchanged outcome does not specify all labels affected by the editor. Subdivision by the gold no-error class retains this same first-error outcome, while the native-utility target uses the metric consequences of the complete edit. Each event set in the table is applied separately to the original Anchor predictions. The total PRMScore gain is computed after applying all accepted edits together.

Table 3: Native consequences of first-error event sets. Each set is applied alone to the matched Anchor; effects are mean PRMScore changes in points and are nonadditive. Counts include repeated target appearances.

Policy	Fixes	Effect	Breaks	Effect	$d = 0$	Effect
PAIR-EF	108497	+3.23	31436	-0.64	206089	+0.83
Paired logistic	108789	+3.24	31497	-0.64	206762	+0.84
Metric-linear	110770	+3.37	37609	-0.56	205174	+1.12

At matched 20% coverage, metric-linear scores 64.89 versus 64.55 for paired logistic; at 40%, it scores 65.56 versus 64.99. These +0.34 and +0.57 increments compare edits at the same intervention frequency, showing that the gain also depends on which edits are selected. The fixed-coverage comparisons and calibration-selected operating points are given in Appendix C.1.

5.3 Model and Calibration-Budget Sensitivity

Table 4 gives the complete Anchor–Critic matrix. For the combined pool, all 15 pair means improve over the original Anchor under PAIR-EF, paired logistic, and metric-linear. Metric-linear gains range from 1.01 to 11.97 points and exceed paired logistic in 13 of the 15 pairs; Skywork–Llama and Skywork–Qwen are the exceptions. The largest gains occur on RLHFlow, which has the lowest Anchor score. Averaging the three Critics within each Anchor gives metric-linear gains from +1.49 for PathFinder to +10.00 for RLHFlow. Excluding RLHFlow leaves mean gains of +2.05 for PAIR-EF and +2.47 for metric-linear, so improvement is also present across the other four Anchors. The reference-only columns repeat within each Anchor because this candidate pool does not depend on the Critic.

Table 4: PRMBench scores by Anchor and Critic, averaged over ten splits. Base is the original Anchor score. EF, Logistic, and Metric use PAIR-EF, paired logistic, and metric-linear with combined candidates; Ref-EF and Ref-Metric use reference-only candidates.

Anchor	Critic	Base	EF	Logistic	Metric	Ref-EF	Ref-Metric
PathFinder	DeepSeek-Math	67.45	68.55	68.58	68.97	68.90	69.08
	Llama-3.1	67.45	68.56	68.49	68.72	68.90	69.08
	Qwen2.5	67.45	68.30	68.26	69.13	68.90	69.08
Qwen2.5-Math-PRM	DeepSeek-Math	65.49	66.07	66.11	66.50	67.40	67.37
	Llama-3.1	65.49	66.56	66.61	67.19	67.40	67.37
	Qwen2.5	65.49	68.30	68.37	69.13	67.40	67.37
ReasonEval	DeepSeek-Math	59.95	63.48	63.46	63.85	64.97	65.15
	Llama-3.1	59.95	63.13	63.18	63.71	64.97	65.15
	Qwen2.5	59.95	65.89	65.89	66.48	64.97	65.15
RLHFlow-Mistral	DeepSeek-Math	54.59	62.26	62.22	63.21	64.34	64.74
	Llama-3.1	54.59	63.29	63.45	63.99	64.34	64.74
	Qwen2.5	54.59	65.77	65.73	66.55	64.34	64.74
Skywork	DeepSeek-Math	64.99	65.66	65.63	66.02	66.51	66.51
	Llama-3.1	64.99	66.24	66.30	66.22	66.51	66.51
	Qwen2.5	64.99	67.46	67.46	67.41	66.51	66.51

Table 5 summarizes budget sensitivity and the second proposal cache. On common target records, metric-linear's gain increases from +2.47 with 100 calibration traces to +3.98 with 1,000. The gain reaches +3.72 at 400 traces and +3.90 at 750, followed by a further 0.07 points at 1,000. All three selectors improve overall with the annotation budget, and metric-linear has the highest mean gain at each tested budget. The smaller increments at the larger budgets show diminishing gains from additional calibration labels over this tested range.

With independently regenerated DeepSeek and Llama proposals, PAIR-EF and paired logistic again perform similarly, at 65.14 and 65.12. Metric-linear reaches 65.46, gaining 2.96 points over the Anchor. The replacement proposals thus reproduce the main selection-target finding: metric-linear has the highest mean score, while subdividing first-error events yields little change relative to

Table 5: Budget sensitivity and replacement-proposal replication on native PRMBench. Left: gains on common target records at each calibration budget. Right: scores and Anchor gains for the second DeepSeek/Llama proposal cache.

Traces	PAIR-EF	Logistic	Metric
100	+2.19	+2.22	+2.47
200	+2.72	+2.72	+3.15
400	+3.20	+3.21	+3.72
750	+3.47	+3.50	+3.90
1000	+3.47	+3.49	+3.98

Selector	Score	Gain
PAIR-EF	65.14	+2.64
Paired logistic	65.12	+2.63
Metric-linear	65.46	+2.96

paired logistic. These results use the same benchmark inputs with newly generated candidates and recalibrated selectors. Appendix E gives the pair-level replacement-proposal results and run summaries.

5.4 Evaluation on DeltaBench

On the adapted DeltaBench correctness task, combined metric-linear improves the mean Anchor score from 52.73 to 56.66 after local calibration, a gain of 3.93 points (Table 6). The gains are +0.33 for Qwen-Math-PRM, +4.94 for Skywork, and +6.52 for RLHFlow. Reference-only metric-linear improves all three Anchor means and reaches 57.17 overall, gaining 4.44 points. In the primary combined-minus-reference comparison, the difference is -0.51 points: +0.26 for Skywork, -1.38 for Qwen-Math-PRM, and -0.40 for RLHFlow. Thus both pools improve the original verification outputs, while reference-only selection has the higher overall mean on this task.

Table 6: DeltaBench correctness-section macro-F1 with metric-linear selection, averaged over ten calibration splits. Δ Base and Δ Ref. compare combined Qwen Critic selection with the original Anchor and reference-only selection.

Anchor	Base	Reference	Combined	Δ Base	Δ Ref.
Qwen-Math-PRM	55.56	57.27	55.89	+0.33	-1.38
Skywork	52.07	56.76	57.01	+4.94	+0.26
RLHFlow	50.57	57.49	57.09	+6.52	-0.40
Mean	52.73	57.17	56.66	+3.93	-0.51

Combined PAIR-EF and paired logistic reach 56.51 and 56.59, while Anchor-only metric-linear reaches 53.54. The three combined-pool selectors are closer here than on PRMBench. Reference-only and combined metric-linear both exceed the Anchor-only mean, with different patterns across Anchors: Anchor-only reaches 57.45 for Qwen-Math-PRM, above both alternatives, while reference and combined candidates give larger gains for Skywork and RLHFlow. The source-by-selector matrix and domain results are given in Appendix F. Together, these comparisons show gains from calibrated editing on the adapted task and locate the differences between candidate sources at the level of individual Anchors.

6 Conclusion

We studied calibrated step-label editing as a way to improve process verification without updating language-model weights. Experiments with PRMs and generative Critics demonstrate gains on native PRMBench and an adapted DeltaBench task. The source comparisons establish reference-PRM candidates as an effective starting point, with additional benefit from Qwen-guided construction on PRMBench. The selection comparisons favor learning native edit utility over finer first-error event categories, and the analysis of accepted edits explains why complete-edit consequences matter. Together, the framework and findings offer a practical route to better verification through the selection and integration of existing model judgments.

AI USE STATEMENT

We used AI tools for drafting and polishing parts of the manuscript. The authors are responsible for the research decisions and the final content of the paper.

REPRODUCIBILITY STATEMENT

The supplementary code includes the model interfaces, grouped calibration procedures, candidate alignment, selector fitting, and evaluation scripts. Configuration files specify the data versions and experimental settings. The scripts generate predictions, per-record confusion counts, and aggregate summaries for the PRMBench and adapted DeltaBench experiments.

REFERENCES

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

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/fef126561bbf9d4467dbb8d27334b8fe-Abstract-Conference.html.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 1321–1330, 2017. URL https://proceedings.mlr.press/v70/guo17a.html.

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

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=IkmD3fKBPQ.

Zeyu Huang, Zihan Qiu, Zili Wang, Edoardo M. Ponti, and Ivan Titov. Post-hoc reward calibration: A case study on length bias. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5d50c76fdf75c24ece568fc84a7125fb-Abstract-Conference.html.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/aca97732e30bcf1303bc22ac3924fd16-Abstract-Conference.html.

David Madras, Toniann Pitassi, and Richard Zemel. Predict responsibly: Improving fairness and accuracy by learning to defer. In *Advances in Neural Information Processing Systems*, 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/hash/09d37c08f7b129e96277388757530c72-Abstract.html.

Hussein Mozannar and David Sontag. Consistent estimators for learning to defer to an expert. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 7076–7087, 2020. URL https://proceedings.mlr.press/v119/mozannar20b.html.

Harikrishna Narasimhan, Rohit Vaish, and Shivani Agarwal. On the statistical consistency of plug-in classifiers for non-decomposable performance measures. In *Advances in Neural Information Processing Systems*, volume 27, 2014. URL https://proceedings.neurips.cc/paper_files/paper/2014/hash/3644e33a5161ec5f3997a6acb98d4447-Abstract.html.

Harikrishna Narasimhan, Purushottam Kar, and Prateek Jain. Optimizing non-decomposable performance measures: A tale of two classes. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pp. 199–208, 2015. URL https://proceedings.mlr.press/v37/narasimhana15.html.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs from preference data. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5503a7c69d48a2f86fc00b3dc09de686-Abstract-Conference.html.

Tej Deep Pala, Panshul Sharma, Amir Zadeh, Chuan Li, and Soujanya Poria. Error typing for smarter rewards: Improving process reward models with error-aware hierarchical supervision. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 14940–14954, 2025. doi: 10.18653/v1/2025.findings-emnlp.807. URL https://aclanthology.org/2025.findings-emnlp.807/.

Young-Jin Park, Kristjan Greenewald, Kaveh Alimohammadi, Hao Wang, and Navid Azizan. Know what you don't know: Uncertainty calibration of process reward models. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/37664246a1e07e212ddacea6e5a523f2-Abstract-Conference.html.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for LLM reasoning. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/98711dea460bdefe0e651ca23ec98ba2-Abstract-Conference.html.

Shuaijie She, Junxiao Liu, Yifeng Liu, Jiajun Chen, Xin Huang, and Shujian Huang. R-PRM: Reasoning-driven process reward modeling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 13438–13451, 2025. doi: 10.18653/v1/2025.emnlp-main.679. URL https://aclanthology.org/2025.emnlp-main.679/.

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. 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)*, pp. 25299–25346, 2025. doi: 10.18653/v1/2025.acl-long.1230. URL https://aclanthology.org/2025.acl-long.1230/.

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

Xinle Wu and Yao Lu. Reward model routing in alignment. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/8dc9b0192713f44f035b8c0bd3f4153a-Abstract-Conference.html.

Shijie Xia, Xuefeng Li, Yixin Liu, Tongshuang Wu, and Pengfei Liu. Evaluating mathematical reasoning beyond accuracy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 27723–27730, 2025. doi: 10.1609/aaai.v39i26.34987. URL https://ojs.aaai.org/index.php/AAAI/article/view/34987.

Zhaohui Yang, Chenghua He, Xiaowen Shi, Shihong Deng, Linjing Li, Qiyue Yin, and Daxin Jiang. Beyond the first error: Process reward models for reflective mathematical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 4711–4728, 2025. doi: 10.18653/v1/2025.findings-emnlp.253. URL https://aclanthology.org/2025.findings-emnlp.253/.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. In *International Conference on Learning Representations*, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/214308a2d5e3f83ef9ad2739e1cbc46d-Abstract-Conference.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, 2025b. 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. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 34932–34940, 2026. 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, 2025. doi: 10.18653/v1/2025.acl-long.50. URL https://aclanthology.org/2025.acl-long.50/.

Yuhang Zhou, Yixin Cao, and Guangnan Ye. From correctness to utility: Gain-based prefix evaluation for LLM reasoning. *arXiv preprint arXiv:2606.07190*, 2026. URL https://arxiv.org/abs/2606.07190.

A Candidate Construction and Selector Fitting

Candidate sources. The direct Critic receives the question and numbered steps. The rerank prompt also includes the three lowest-scoring steps and their scores from the shared Qwen2.5-Math-PRM-7B reference PRM. These reference scores use the converted input representation; editing operates on each Anchor's native labels. Candidate prompts and proposals are shared across Anchors and seeds. In the PRMBench combined pool, direct parsing failures supply the stored no-error fallback. Direct-only excludes parsing failures; reference-only offers the lowest-reference-score step and a no-error action, without using a reference threshold or Critic decision. The selector decides whether to apply either candidate.

Rerank routing. All three Critics use a fixed reference-score threshold of 0.5. The rerank branch supplies a Critic proposal when the minimum reference score is below 0.5 and a no-error candidate otherwise. This rule is applied when constructing the evaluated candidate pool, independently of the routing decisions stored with the original proposals. Direct proposals remain available. All three Critics have a parseable rerank response for every trace selected by the common rule.

Shared fitting choices. All logistic selectors use zero-initialized Newton fitting for at most 20 iterations, stopping when the update norm is below 10^{-6}. Rare and zero-positive event heads use the same fitting rule and an unpenalized intercept, without event pseudocounts or rebalancing. The linear regressor uses a direct ridge solve with an unpenalized intercept. Event-head reliability and zero-positive fit counts are given in the supplementary results. Feature quantiles use right-inclusive ranks over native validity scores from fit traces. No-error actions have normalized position zero; exact score ties prefer the direct candidate to the reranked candidate. Anchor-only controls append indicators for the lowest-score-step and no-error actions, set to zero outside that pool.

B Statistical Analysis

PRMBench point estimates average 15 model pairs and ten splits equally. The 95% intervals use 2,000 bootstrap replicates of source-question groups, with shared weights across models and split seeds and analysis seed 2027. Each replicate recomputes the native metric from weighted confusion counts. Intervals describe question variability conditional on fitted policies and evaluated models; repeated appearances of a question are not independent observations. Split signs separately describe partition sensitivity. Table 7 gives paired candidate-source and selector contrasts. The combined metric-linear versus Anchor-only PAIR-EF row changes both the candidate pool and selector; the same-selector rows isolate the candidate-source comparison.

Table 7: Paired PRMBench score contrasts in points. Intervals resample source-question groups jointly across model pairs and splits. Combined denotes direct and reranked Critic candidates; EF, Logistic, and Metric denote PAIR-EF, paired logistic, and metric-linear.

Contrast	Difference	95% interval
Combined EF − Anchor-only EF	+1.45	[+1.29, +1.63]
Combined Metric − Anchor-only EF	+1.96	[+1.79, +2.12]
Combined EF − Reference-only EF	-0.45	[-0.66, -0.24]
Combined Metric − Reference-only Metric	-0.10	[-0.30, +0.12]
Direct-only EF − Anchor-only EF	+0.32	[+0.15, +0.48]
Combined EF − Combined Logistic	-0.014	[-0.024, -0.004]
Combined Metric − Combined Logistic	+0.49	[+0.43, +0.55]

C Additional Selection Comparisons

Baseline comparisons. Table 8 collects the candidate-edit baselines with paired intervals, coverage, and the number of runs with negative gains. Direct and rerank edits use the shared editor. Whole deferral selects one of keeping the Anchor, direct editing, or rerank editing on the tuning partition and applies that strategy across the target set. All scores evaluate the resulting Anchor step-label vectors. Learned Anchor-only selection reaches 64.24–64.52 across the three selectors in Table 8.

Table 8: PRMBench baselines with 1,000 calibration traces. Gains and paired question-group intervals are in points. Neg. counts negative gains over the original Anchor among 150 Anchor–Critic–split runs. Coverage is the percentage of target records edited.

Policy	Score	Gain	95% interval	Neg.	Coverage
Anchor	62.49	+0.00	[+0.00, +0.00]	0	0.00
Anchor-only PAIR-EF	64.52	+2.02	[+1.86, +2.19]	0	32.00
Anchor-only paired logistic	64.24	+1.74	[+1.59, +1.90]	0	32.66
Anchor-only metric-linear	64.46	+1.97	[+1.80, +2.15]	9	34.70
Reference-only PAIR-EF	66.42	+3.93	[+3.67, +4.17]	0	42.93
Reference-only paired logistic	66.45	+3.95	[+3.69, +4.20]	0	43.53
Reference-only metric-linear	66.57	+4.08	[+3.82, +4.33]	0	43.61
PAIR-EF	65.97	+3.47	[+3.31, +3.63]	1	38.65
Paired logistic	65.98	+3.49	[+3.33, +3.65]	1	38.76
Metric-linear	66.47	+3.98	[+3.81, +4.15]	0	39.49
Confidence gate	64.93	+2.44	[+2.29, +2.59]	23	33.63
Direct edit	61.09	-1.41	[-1.72, -1.11]	84	67.53
Rerank edit	63.74	+1.24	[+0.97, +1.52]	85	48.39
Whole deferral	64.91	+2.42	[+2.27, +2.56]	14	34.44

Global score-threshold retuning with the option to retain the Anchor reaches 63.70. Unconditional lowest-score-step and no-error edits score 60.99 and 46.31, respectively. Learned selection achieves a higher average score than these controls.

Selection-target and acceptance ablations. The full learned-selector ablations use the same 1,000-trace calibration partitions and candidate pool. "No threshold" accepts the highest-ranked candidate on every trace that has an action. Error-only and no-error-only retain the corresponding two heads; fix-only and break-avoid retain only the indicated event sign. The metric-weighted event ablation weights each fitted event probability by that event's mean fit-set marginal native utility. Empty event groups use the corresponding fix- or break-group mean, falling back to zero if that sign is absent. This assigns one average utility to each fix or break category; metric-linear learns from the utility of each candidate, including $d = 0$ edits. In Table 9, metric-weighted events score 66.00, close to paired logistic at 65.98. Metric-linear reaches 66.47. The no-threshold score of 62.34 versus 65.97 for tuned PAIR-EF shows the contribution of the acceptance threshold under the same ranking rule.

Table 9: Selection-target and acceptance ablations with combined candidates and 1,000 calibration traces. Gain is relative to the original Anchor. Negative runs count negative gains among 150 Anchor–Critic–split runs; coverage is in percent.

Policy	Score	Gain	Negative runs	Coverage
PAIR-EF	65.97	+3.47	1	38.65
Paired logistic	65.98	+3.49	1	38.76
Metric-weighted events	66.00	+3.50	3	38.63
Metric-linear	66.47	+3.98	0	39.49
Error-only	65.90	+3.41	2	38.74
No-error-only	64.29	+1.80	22	30.69
Fix-only	65.51	+3.02	15	37.74
Break-avoid	65.85	+3.35	0	38.38
No threshold	62.34	-0.16	71	75.07

Matched-learner supervision comparison. Table 2 changes only the target of the ridge regressor. The correct-step target is $(TP_a + TN_a) - (TP_0 + TN_0)$, using the same evaluated label channels as the native metric. All three targets use identical fit records, features, ridge coefficient 1, and tuning rules. Native utility improves over first-error utility by 0.21 points on PRMBench and over correct-step-count change by 0.87 points. This comparison supports metric-aware supervision beyond changing the learner. On DeltaBench, native utility exceeds correct-step-count change by 2.13 points; its 0.16-point difference from first-error utility is smaller and its paired interval includes zero. The intervals use shared question-group bootstrap weights across models and splits: 2,000 replicates for PRMBench and 20,000 for DeltaBench, with seed 2027.

C.1 Calibration-selected operating points

Table 10 compares fixed target-coverage points as a descriptive ranking diagnostic. The deployment comparisons that follow choose their policies from calibration data. At both displayed coverage levels, metric-linear has the highest PRMScore and a higher first-error break ratio than paired logistic. These metrics capture different aspects of the selected edits, consistent with the analysis in Table 3.

Table 10: Diagnostic native PRMScore at fixed target coverage. Entries show score, gain over the Anchor, and break rate $B/(F + B)$.

Ranker	20% coverage	40% coverage
PAIR-EF	64.55 (+2.05; 14.2%)	64.96 (+2.47; 30.3%)
Paired logistic	64.55 (+2.05; 14.6%)	64.99 (+2.49; 30.2%)
Metric-linear	**64.89** (+2.40; 19.8%)	**65.56** (+3.07; 32.8%)
Confidence gate	62.42 (-0.07; 50.5%)	63.39 (+0.89; 46.8%)

Optional group-level utility check. After tuning, GLCB evaluates the selected risk-partition interventions. For n selected events in G source-question groups, set

$$\widehat{\mathrm{se}}(\bar{d}) = \frac{1}{n}\sqrt{\frac{G}{G-1}\sum_{g}\left[\sum_{i\in g}(d_i - \bar{d})\right]^2}. \tag{6}$$

The tuned policy is deployed if $G \geq 5$ and $\bar{d} - \Phi^{-1}(0.90)\widehat{\mathrm{se}}(\bar{d}) \geq -0.02$; otherwise the Anchor is retained. GLCB is a cluster-normal approximation to mean exact-correctness utility, applied identically to all rankers. It is distinct from conformal risk control (Angelopoulos et al., 2024); its behavior is assessed through target outcomes.

The informative-event break ratio is $\theta = \Pr(d = -1 \mid d \neq 0)$, with reference level $\rho = 0.35$. For a fixed intervention population, $\mathbb{E}[d] = \Pr(d \neq 0)(1 - 2\theta)$, but neither quantity determines native PRMScore. We report $B/(F + B)$ and exceedances above 0.35 among informative target runs; zero-event runs have an undefined ratio. The GLCB mean-utility condition and the break-ratio reference therefore measure different consequences.

Table 11 applies the GLCB check to the combined pool. Adding GLCB to metric-linear changes PRMScore from 66.47 to 66.02 and coverage from 39.49% to 30.63%. The empirical break ratio falls from 25.3% to 19.9%, and exceedances fall from 57/149 to 14/104 informative runs. Thus the optional check changes the score–coverage trade off. The reported exceedances use the separate break-ratio reference of 0.35; GLCB decides whether to deploy the editing policy using a lower confidence bound on the average change in first-error correctness.

Table 11: Calibration-selected deployment points. Coverage and break ratio are percentages. Risk pass counts the risk-partition checks that pass; only GLCB rows enforce that check. Exceedances above 0.35 use informative target-run denominators, excluding runs without fixes or breaks.

Policy	Risk pass	Score	Coverage	$B/(F+B)$	Exceed.
PAIR-EF	125/150	65.97	38.65	22.5	32/150
PAIR-EF + GLCB	125/150	65.81	33.13	19.2	11/125
Paired logistic	126/150	65.98	38.76	22.5	32/150
Paired logistic + GLCB	126/150	65.83	33.95	19.9	13/126
Metric-linear	104/150	66.47	39.49	25.3	57/149
Metric-linear + GLCB	104/150	66.02	30.63	19.9	14/104
Confidence gate	61/150	64.93	33.63	33.9	57/106
Confidence gate + GLCB	61/150	64.85	23.34	26.4	14/61

D Proposal Coverage and Edit Outcomes

Table 12 separates proposal availability from the outcomes of accepted edits. Original correct controls and other unmatched records retain the Anchor predictions because they have no compatible proposals. The modified traces with no annotated errors do receive edits. For metric-linear, these produce 2,169 first-error fixes and 3,292 breaks across repeated target appearances. The original-control rows describe retained predictions; the matched no-error rows describe actual edit outcomes.

Table 12: Combined-pool coverage and first-error outcomes across 150 target runs. Appearances count traces repeatedly across model pairs and splits. No-error labels include original controls and modified traces with an empty error set. Each stratum is summarized separately and may overlap another.

Policy	Stratum	Appearances	Selected	Fix	Break
PAIR-EF	Original correct	97320	0	0	0
PAIR-EF	All no-error labels	117825	6609	1578	4493
PAIR-EF	Error labels	777480	339413	106919	26943
PAIR-EF	Unavailable proposals	99705	0	0	0
Paired logistic	Original correct	97320	0	0	0
Paired logistic	All no-error labels	117825	6633	1623	4470
Paired logistic	Error labels	777480	340415	107166	27027
Paired logistic	Unavailable proposals	99705	0	0	0
Metric-linear	Original correct	97320	0	0	0
Metric-linear	All no-error labels	117825	6267	2169	3292
Metric-linear	Error labels	777480	347286	108601	34317
Metric-linear	Unavailable proposals	99705	0	0	0

E Replication with Regenerated Proposals

This replication uses independently regenerated PRMBench proposals from DeepSeek-Math and Llama-3.1. Features, fitting rules, the 1,000-trace budget, and grouped splits are shared with the primary experiment. Selector parameters and the operating threshold are recalibrated for the replacement proposals.

Table 13: PRMBench replication with regenerated proposals. Top: pair means over ten splits. Bottom: aggregate scores, positive/negative/zero-gain runs out of 100, and mean coverage. Parentheses give gains over the corresponding Anchor; the mean Anchor score is 62.49. Logistic, Metric, and Confidence denote paired logistic, metric-linear, and confidence gating.

Anchor	Critic	Base	PAIR-EF	Logistic	Metric	Confidence
PathFinder	DeepSeek-Math	67.45	68.40 (+0.95)	68.43 (+0.98)	68.84 (+1.39)	67.00 (-0.45)
PathFinder	Llama-3.1	67.45	68.53 (+1.09)	68.50 (+1.05)	68.56 (+1.12)	67.32 (-0.12)
Qwen-Math-PRM	DeepSeek-Math	65.49	65.86 (+0.36)	65.81 (+0.31)	66.39 (+0.90)	65.40 (-0.09)
Qwen-Math-PRM	Llama-3.1	65.49	66.25 (+0.75)	66.15 (+0.65)	66.52 (+1.03)	65.21 (-0.28)
ReasonEval	DeepSeek-Math	59.95	62.76 (+2.81)	62.72 (+2.77)	63.24 (+3.29)	62.43 (+2.48)
ReasonEval	Llama-3.1	59.95	63.40 (+3.45)	63.39 (+3.45)	63.65 (+3.70)	61.99 (+2.04)
RLHFlow-Mistral	DeepSeek-Math	54.59	61.65 (+7.06)	61.64 (+7.05)	62.26 (+7.67)	60.18 (+5.59)
RLHFlow-Mistral	Llama-3.1	54.59	62.76 (+8.17)	62.77 (+8.18)	63.10 (+8.51)	61.60 (+7.01)
Skywork	DeepSeek-Math	64.99	65.64 (+0.64)	65.72 (+0.73)	65.93 (+0.93)	64.78 (-0.22)
Skywork	Llama-3.1	64.99	66.15 (+1.16)	66.13 (+1.13)	66.10 (+1.10)	64.33 (-0.67)

	PAIR-EF	Logistic	Metric	Confidence	Metric-weighted events	PAIR-EF + GLCB
Score (gain)	65.14 (+2.64)	65.12 (+2.63)	65.46 (+2.96)	64.02 (+1.53)	65.15 (+2.66)	64.96 (+2.46)
Runs $+/-/0$	97/3/0	97/2/1	98/1/1	42/32/26	97/2/1	78/1/21
Coverage (%)	32.68	33.74	32.95	26.42	33.57	26.99

All ten pair means improve over their Anchor for PAIR-EF, paired logistic, and metric-linear. Metric-linear exceeds paired logistic in nine pairs; Skywork–Llama is the exception. Confidence gating has negative mean gains for both Critics on PathFinder, Qwen-Math-PRM, and Skywork. The regenerated proposals preserve the close mean scores of PAIR-EF and paired logistic and the higher overall score of metric-linear.

F Detailed Analysis on DeltaBench

The supplied section boundaries define both the inference and evaluation units. Only question text and section content enter model inputs; annotation explanations, section descriptions, usefulness labels, and reference answers are excluded. After removing 137 traces whose questions overlap with prior data, the full-input context filter retains 468 of 1,099 traces (42.58%), or 37.86% of the public release. The other 631 traces exceed at least one model's context limit. Of the eligible traces, mathematics contributes 71, coding 195, general reasoning 88, and physics/chemistry/biology 114. The target domain counts are reported in Table 15.

Candidate coverage and consequences. Every eligible trace has predictions from the three Anchors, reference scores, and direct Critic proposals. Unparseable or out-of-range Critic proposals are omitted rather than replaced with reference predictions. Non-routed reranking supplies a no-error candidate. The same Critic and reference outputs are shared across Anchors and calibration splits. The target includes six traces with no reasoning-correctness errors. Table 16 reports the edits that introduce invalid labels on these traces and, separately, label changes across all accepted $d = 0$ edits. For Skywork, the editing policy increases false alarms on all six fully correct target traces. The corresponding counts average 0.90 traces for Qwen-Math-PRM and zero for RLHFlow over ten calibration splits. Skywork's overall score improvement is thus accompanied by increased false alarms on this subset.

Candidate-source and domain results. Table 14 expands the main comparison to all four candidate pools and selectors. For Qwen-Math-PRM, Anchor-only metric-linear scores 57.45, above reference-only at 57.27 and combined at 55.89. For Skywork and RLHFlow, the reference-only and combined pools score higher than Anchor-only selection under each of the three learned selectors. Table 15 evaluates the reference-only and combined metric-linear policies within each target domain. Both improve the mean Anchor score in all four domains. The combined-minus-reference difference is +1.54 points for science and -2.22 for coding, showing variation in the candidate-source comparison across domains.

Table 14: Candidate-source and selector comparison on the adapted DeltaBench task. Scores are section macro-F1 averaged over ten calibration splits, using the same 117 calibration and 351 target traces. EF, Logistic, Metric, and Confidence denote PAIR-EF, paired logistic, metric-linear, and confidence gating.

Anchor	Pool	EF	Logistic	Metric	Confidence
Qwen-Math-PRM	Anchor only	56.34	56.28	57.45	55.82
	Direct only	55.21	55.08	55.63	55.17
	Reference only	56.33	56.28	57.27	55.92
	Combined	56.23	56.16	55.89	55.92
Skywork	Anchor only	52.49	52.49	52.44	52.38
	Direct only	52.96	52.99	53.03	52.62
	Reference only	57.11	57.15	56.76	56.54
	Combined	56.40	56.76	57.01	55.13
RLHFlow	Anchor only	50.24	50.26	50.73	50.43
	Direct only	52.41	52.48	52.61	52.48
	Reference only	57.16	57.07	57.49	57.55
	Combined	56.90	56.84	57.09	56.09

Table 15: DeltaBench section macro-F1 by domain with metric-linear selection. Each row recomputes the score within the domain and averages three Anchors and ten splits. Δ Ref. compares combined and reference-only candidates. The pooled score is computed from confusion counts over the full target set.

Domain	Traces	Base	Reference	Combined	Δ Ref.
Mathematics	47	52.30	54.18	53.64	-0.53
Coding	150	52.22	58.33	56.11	-2.22
General	67	53.40	59.80	58.96	-0.84
Physics/chemistry/biology	87	51.64	55.24	56.77	+1.54

Table 16: Accepted-edit outcomes for combined metric-linear on DeltaBench, averaged over ten splits per Anchor. Damaged correct and added-invalid counts measure increases in invalid predictions over the Anchor on the six fully correct target traces. The final columns count accepted $d = 0$ edits and sections whose predicted labels change.

Anchor	Cov. (%)	Damaged correct	Added invalid	$d = 0$ edits	Changed sections
Qwen-Math-PRM	40.54	0.90	0.90	89.30	176.40
Skywork	69.00	6.00	6.00	161.60	183.40
RLHFlow	71.97	0.00	0.00	175.80	323.00

G Qualitative Examples of Label Editing

These DeltaBench target examples use the Qwen2.5-7B-Instruct Critic. Excerpts retain their original section numbers. V/I denote valid/invalid; "matched / total" counts agreement with benchmark step labels. "Correct" means complete label agreement; otherwise the outcome is "Wrong". A miss labels an invalid step valid; a false alarm labels a valid step invalid. Parentheses identify the affected steps. Reference-only uses metric-linear; the other learned selectors use combined candidates. Direct edit applies the direct proposal through the same editor.

G.1 TEMPLATE SUBSTITUTION: REPAIR AND RETENTION

Source: FullStackBench via DeltaBench. **Anchor:** Skywork-PRM-7B, split seed 0.

Question. Write a function named templateEngine in JS, which takes a template string and a data object as its parameters. The keys enclosed in curly braces {} in the template string will be replaced by the corresponding values from the data object.

Original output excerpts. **Step 1.** I need to write a JavaScript function called `templateEngine` that accepts two arguments: `templateString` (a string potentially containing placeholders like {`key`}) and `dataObject` (an object whose keys correspond to the placeholders).

Step 2. The function should iterate through the `templateString`, identify the placeholders, and replace them with the values from `dataObject`.

Step 3.

```
function templateEngine(templateString, dataObject) {
  return templateString.replace(/\{\{(.*?)\}\}/g, (match, key) => {
    return dataObject[key.trim()] !== undefined ? dataObject[key.trim()] :
    match;
  });
}
```

The double-brace pattern fails to replace the required single-brace placeholders.

Recorded proposals and edits. The direct Critic proposes no error; reranking proposes step 3. The latter clears Skywork's step-2 false alarm and identifies the code error.

Judgment	Selected action	Step labels	Matched / total	Outcome / errors
Anchor	Keep	V I V	1/3	Wrong: miss (3); false alarm (2)
Direct edit	No error	V V V	2/3	Wrong: miss (3)
Reference-only	Keep	V I V	1/3	Wrong: miss (3); false alarm (2)
PAIR-EF	Step 3	V V I	3/3	Correct
Paired logistic	Step 3	V V I	3/3	Correct
Metric-linear	Step 3	V V I	3/3	Correct
Benchmark labels	—	V V I	3/3	Reference

For Qwen2.5-Math-PRM-7B (seed 1), all three learned selectors retain the already correct `V V I` vector. Applying the same no-error proposal would reduce matched labels from 3/3 to 2/3.

G.2 CPU MONITORING: COMPARING TWO CANDIDATE EDITS

Source: BigCodeBench via DeltaBench. **Anchor:** Qwen2.5-Math-PRM-7B, split seed 1.

Question. Monitors and logs CPU usage at specified intervals over a given duration. Note that: Actual run time of the function may slightly exceed the specified 'duration' due to processing time and system response delay. The function records the CPU usage percentage at regular intervals for a specified duration. The data is captured every 'interval' seconds until the 'duration' is reached or exceeded. Each record includes a timestamp and the CPU usage percentage at that moment. The data is saved in JSON format in a log file named 'logfile.log'. The function supports different commands for CPU usage monitoring on Windows and Unix/Linux platforms. The function should raise the exception for: ValueError: If either 'interval' or 'duration' is less than or equal to zero. The function should output with: str: Path to the log file where CPU usage data is saved. Returns None if an IOError occurs during file operations. You should write self-contained code starting with:

```
import subprocess
import time
import json
import platform
LOGFILE_PATH = "logfile.log"
def task_func(interval, duration):
```

Original output excerpts. **Step 1.** The user wants a Python function `task_func` that monitors CPU usage and logs it to a JSON file.

Step 2. Alternative Linux command using `mpstat`:

```
mpstat 1 1 | awk '$3 == "all" { print 100-$NF }'
```

Let's choose `mpstat` as it's cleaner. We'll need to check if it's installed. If not, we might revert to `top`.

Step 3. Record timestamp and usage.

Step 4. The JSON output should be a list of dictionaries.

Step 5. Let's start writing the code. The implementation includes the following consecutive lines:

```
data = []
    start_time = time.time()
    end_time = start_time + duration
```

The unexpected indentation at `start_time` makes the step-5 implementation invalid.

Recorded proposals and edits. The no-error direct proposal clears the step-2 false alarm. The reranked step-5 proposal also detects the implementation error.

Judgment	Selected action	Step labels	Matched / total	Outcome / errors
Anchor	Keep	V I V V V	3/5	Wrong: miss (5); false alarm (2)
Direct edit	No error	V V V V V	4/5	Wrong: miss (5)
Reference-only	Keep	V I V V V	3/5	Wrong: miss (5); false alarm (2)
PAIR-EF	Step 5	V V V V I	5/5	Correct
Paired logistic	Step 5	V V V V I	5/5	Correct
Metric-linear	Step 5	V V V V I	5/5	Correct
Benchmark labels	—	V V V V I	5/5	Reference

G.3 Partial Improvement without First-Error Recovery

Source: BigCodeBench via DeltaBench. **Anchor:** RLHFlow/Llama3.1-8B-PRM-Mistral-Data, split seed 0.

Question. Solve a quadratic equation in the form of ax ^ 2 + bx + c = 0, where a, b, and c randomly generated numbers are between -10 and 10. The solutions are complex numbers rounded to the specified accuracy. The function should output with: tuple: A tuple of two solutions formatted as complex numbers (rounded to the specified precision). You should write self-contained code starting with:

```
import numpy as np
from sympy import symbols, solve
def task_func(precision=2, seed=0):
```

Original output excerpts. **Step 1.** The user wants a Python function that solves quadratic equations with random coefficients. The coefficients `a`, `b`, and `c` should be integers between -10 and 10 (inclusive). The solutions can be complex numbers and need to be rounded to the specified precision. The function should return a tuple of the two solutions.

Step 2. Calculate the discriminant: Calculate the discriminant (delta) of the quadratic equation: delta = b^2 - 4ac.

Step 3. Let's implement the function step-by-step. The original implementation contains:

```
delta = b**2 - 4 * a * c

if delta >= 0:
        x1 = (-b + np.sqrt(delta)) / (2 * a)
        x2 = (-b - np.sqrt(delta)) / (2 * a)
        sol1 = complex(round(x1, precision))
        sol2 = complex(round(x2, precision))
    else:
        real_part = -b / (2 * a)
        imaginary_part = np.sqrt(abs(delta)) / (2 * a)
```

The misaligned `else` prevents the implementation from parsing.

Recorded proposals and edits. The direct Critic proposes no error. This record is not routed to reranking, so the candidate pool contains only that action. Metric-linear accepts it, removing the step-1 false alarm; PAIR-EF and paired logistic retain the Anchor.

Judgment	Selected action	Step labels	Matched / total	Outcome / errors
Anchor	Keep	`I V V`	1/3	Wrong: miss (3); false alarm (1)
Direct edit	No error	`V V V`	2/3	Wrong: miss (3)
Reference-only	Step 3	`V V I`	3/3	Correct
PAIR-EF	Keep	`I V V`	1/3	Wrong: miss (3); false alarm (1)
Paired logistic	Keep	`I V V`	1/3	Wrong: miss (3); false alarm (1)
Metric-linear	No error	`V V V`	2/3	Wrong: miss (3)
Benchmark labels	—	`V V I`	3/3	Reference

Effect on the two targets. The Anchor's first-error prediction (step 1) and the accepted proposal (no error) both disagree with the benchmark's step 3, giving $d = 0$. Nevertheless, one previously incorrect step label becomes correct. This is a partial improvement: the accepted edit still misses the code error, whereas the reference-only policy selects step 3 and recovers all three labels.

G.4 A MISLOCALIZED EDIT

Source: Beginning and Intermediate Algebra via DeltaBench. **Anchor:** Skywork-PRM-7B, split seed 0.

Question. Simplify the radical expression

$$-6\sqrt[4]{80m^4p^7q^4}.$$

Original output excerpts. **Step 2.** 80 is $2^4 \times 5$.

Step 3.

$$\sqrt[4]{p^7} = p\sqrt[4]{p^3}.$$

Step 4.

$$\sqrt[4]{80m^4p^7q^4} = 2mq\sqrt[4]{5p^3}.$$

Step 6.

$$\boxed{-12mq\sqrt[4]{5p^3}}$$

Step 4 drops the factor p extracted in step 3. The benchmark marks step 4 invalid. Step 2's numerical factorization is correct.

Recorded proposals and edits. The direct Critic proposes step 2. This record is not routed to reranking; the alternative no-error action matches the Anchor's action and is removed. All three learned selectors accept step 2, while reference-only selects step 5.

Judgment	Selected action	Step labels	Matched / total	Outcome / errors
Anchor	Keep	V V V V V V	5/6	Wrong: miss (4)
Direct edit	Step 2	V I V V V V	4/6	Wrong: miss (4); false alarm (2)
Reference-only	Step 5	V V V V I V	4/6	Wrong: miss (4); false alarm (5)
PAIR-EF	Step 2	V I V V V V	4/6	Wrong: miss (4); false alarm (2)
Paired logistic	Step 2	V I V V V V	4/6	Wrong: miss (4); false alarm (2)
Metric-linear	Step 2	V I V V V V	4/6	Wrong: miss (4); false alarm (2)
Benchmark labels	—	V V V I V V	6/6	Reference

Editing consequence. The original vector misses the step-4 error but introduces no false alarms. The accepted step-2 edit adds a false alarm and leaves that error undetected, reducing matched labels from 5/6 to 4/6. First-error correctness is unchanged ($d = 0$), just as in Appendix G.3, but the step-level effect has the opposite sign. Together, the two cases make the limitation of a first-error-only target concrete: unchanged localization correctness can accompany either improved or degraded step judgments.