TRACING ERROR ORIGINS AND PROPAGATION: DEPENDENCY-GROUNDED PROCESS SUPERVISION FOR MATHEMATICS AND PHYSICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Mathematical and physical reasoning is central to AI for Science, and process reward models help improve it by evaluating intermediate steps. Derivations in these domains reuse quantities, equations, and assumptions, creating identifiable dependencies between the information a step uses and the results it produces. These dependencies provide a basis for distinguishing newly introduced mistakes from inherited errors. We propose dependency-grounded process supervision, which extracts these input–output relationships and checks whether each error arises locally or is inherited through the information used. The resulting supervision teaches a model to judge step correctness, distinguish new mistakes from inherited consequences, and classify newly introduced mistakes. This labeling preserves valid uses of information from partially incorrect steps and accounts for steps that both inherit an error and introduce another. Experiments in mathematics and physics show improved error detection compared with models trained to judge step correctness, with gains persisting on longer mathematical solutions. Controlled comparisons show that learning to distinguish new and propagated errors improves diagnosis beyond rules based on correctness and step order, with similar detection performance. The learned scores also support answer selection and training-data selection. These findings suggest that reasoning dependencies can guide the design of more informative feedback about how errors arise and propagate in scientific problem solving.

1 INTRODUCTION

Improving mathematical and physical reasoning is an important objective for AI for Science. Solving problems in these domains requires connecting assumptions, intermediate quantities, and derived conclusions through valid steps. Recent LLM advances use supervised fine-tuning, reinforcement learning, and reward-guided data selection and inference (Yang et al., 2024; DeepSeek-AI et al., 2025). These approaches depend on feedback about reasoning quality. A final-answer reward does not identify which step caused a failure or certify every intermediate inference. Process reward models (PRMs) provide localized feedback for evaluating and selecting reasoning traces (Lightman et al., 2023).

BFE extends process supervision beyond the first error by labeling propagation and cessation throughout reflective solutions (Yang et al., 2025); PathFinder-PRM conditions rewards on mathematical and consistency error types (Pala et al., 2025). Generative verifiers provide reasoning about step validity (Khalifa et al., 2025) and physics-specific critiques (Dong et al., 2026), while PRISM-Physics evaluates derivations against structured reference formulas (Zhao et al., 2026a). These advances enrich process feedback, motivating a closer connection between supervision targets and the dependencies within a derivation.

Mathematical and physical derivations offer structure for this connection: beyond explicit lemma applications, steps reuse quantities, equations, and assumptions established earlier. Under stated conditions, these input–output links make dependencies identifiable within a solution. They provide evidence for distinguishing a new mistake from inherited invalidity. A valid calculation using an erroneous quantity can produce a wrong conclusion without introducing a new mistake. Conversely,

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.

an incorrect source step may also supply a valid definition used elsewhere, as in our thin-film example (Figure 2). A step can also inherit an error and introduce another. Assigning supervision in these cases requires identifying *which source content is used and what the current step does with it*.

We introduce dependency-grounded process supervision to make these distinctions explicit. Input–output extraction links each step to the quantities and premises it consumes, preserving the original reasoning text. Attribution then separates contextual correctness, error role, and intrinsic type. All evaluable steps receive correctness supervision; intrinsic types describe newly introduced mistakes, including those that coexist with inherited errors. A PRM trained with these targets supplies scores for answer selection and training-data selection.

We evaluate detection and diagnosis in mathematics and physics and investigate whether supervision gains increase with reasoning length. Shared-label comparisons and downstream answer and data selection connect supervision design to problem solving. Our contributions are:

1. **Error attribution through consumed inputs.** Step input–output references distinguish new mistakes from propagated consequences while preserving valid uses of partially incorrect steps.
2. **Attribution-derived supervision.** Correctness, role, and intrinsic-type targets retain full-trace feedback and supervise the new mistake in mixed errors, learned by a PRM.
3. **Diagnostic and downstream evaluation.** Shared-label comparisons measure role prediction beyond correctness-based sequential inference. Mathematics and physics evaluations assess detection and diagnosis, complemented by reasoning-length analyses, answer voting, and SFT data selection.

2 Related Work

Error localization, propagation, and typing. Step correctness supports process supervision (Lightman et al., 2023), localization (Zheng et al., 2025), and repair (Zhang et al., 2026a). BFE assigns binary labels throughout reflective traces using propagation and cessation rules (Yang et al., 2025). PathFinder-PRM predicts mathematical and consistency errors, then conditions reward prediction on those types (Pala et al., 2025). Our attribution resolves consumed inputs to source outputs and separates inherited invalidity from newly introduced mistakes. This determines role and intrinsic-type targets, including mixed errors, while preserving valid outputs of incorrect sources. Error categories alone do not identify the upstream content responsible for invalidity.

Dependencies for verification and credit assignment. Natural Program verifies deductions against premises (Ling et al., 2023); PRISM-Physics uses reference formula DAGs and symbolic matching (Zhao et al., 2026a). VeryTrace compiles dependencies for verification and repair (Zhong et al., 2026); Graph-GRPO uses dependency graphs for reward credit assignment (Che et al., 2026).

Verification and reward modeling. ThinkPRM learns verification reasoning from limited labels (Khalifa et al., 2025); WebArbiter generates principle-guided justifications and preferences for web agents (Zhang et al., 2026b). CRM links conditional step rewards to final outcomes (Zhang et al., 2025); SPRO derives policy-based process rewards and step-wise advantages for reinforcement learning (Fei et al., 2025). Their credit signals guide trajectory selection or policy optimization. Our attribution instead specifies the semantic supervision target: which mistake a step introduces and which invalid content it inherits. These distinctions determine role labels and intrinsic-type eligibility. PhysPRM provides physics critiques and error types for selection and refinement (Dong et al., 2026); Sci-PRM supervises scientific tool use (Zhao et al., 2026b). PRM label imbalance (Agrawal et al., 2026) motivates our sparse-error analysis.

3 Dependency-Grounded Process Supervision

Given a problem x and its normalized solution text R, we construct steps $S = (s_1, \ldots, s_n)$ and supervision that distinguishes an error introduced at a step from invalidity inherited through its inputs. Each step has contextual correctness $c_i \in \{\mathsf{C}, \mathsf{I}, \mathsf{U}\}$, where U denotes insufficient evidence.

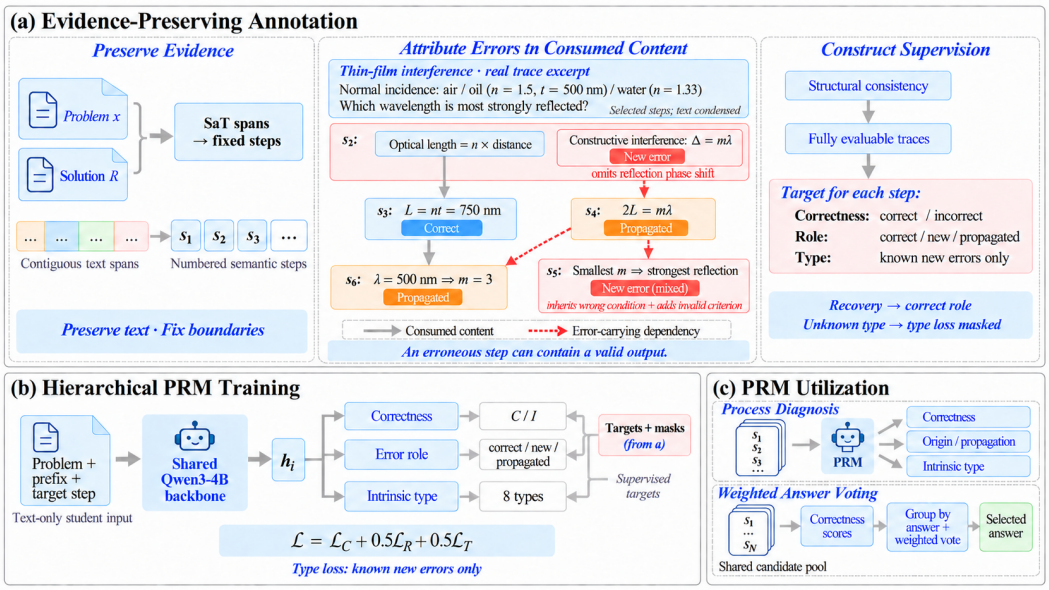


Figure 1: Overview of dependency-grounded process supervision. Consumed-content dependencies support error attribution and hierarchical targets for correctness, role, and intrinsic type. These annotations train a text-prefix PRM for process diagnosis and weighted answer voting. Gray arrows denote dependencies; dashed red arrows identify error-carrying dependencies.

The prediction task is

$$z_i = (x, \widetilde{s}_{<i}, s_i), \qquad f_\theta(z_i) = (p_i^C, p_i^R, p_i^T), \tag{1}$$

where $\widetilde{s}_{<i}$ is the history retained within the context budget, and the outputs predict correctness, error role, and intrinsic type. Dependency evidence constructs the supervision targets; the PRM evaluates each step using the problem and preceding reasoning. Figure 1 summarizes annotation, PRM training, and downstream use.

3.1 Input–Output Extraction and Dependencies

When a source step contains valid and invalid content, attribution must identify which output a consumer uses. We extract these references under fixed, text-preserving step boundaries. SaT-3L-SM (Frohmann et al., 2024) proposes candidate spans $(a_1, \ldots, a_m)$, which the compiler groups into reasoning steps:

$$0 = b_0 < \cdots < b_n = m, \qquad s_i = \Big\Vert_{j=b_{i-1}+1}^{b_i} a_j, \qquad R = \Big\Vert_{i=1}^{n} s_i. \tag{2}$$

This partition fixes the unit of judgment without rewriting the solution or correcting its errors.

For each step, the compiler extracts its operation, consumed inputs D_i, newly adopted assumptions $\mathcal{A}_i$, and atomic outputs $\mathcal{O}_i$. Writing $\mathcal{Q}$ for the facts and conditions supplied by the problem, valid input references satisfy

$$D_i \subseteq \mathcal{Q} \cup \bigcup_{j<i} (\mathcal{O}_j \cup \mathcal{A}_j). \tag{3}$$

Resolving each input to a problem fact or an earlier output or assumption establishes its direct dependency. References identify individual outputs, so a consumer can use an unaffected output from an incorrect step. This representation supplies the evidence for error inheritance. Extraction preserves errors without judging correctness; attribution checks relevant references under the fixed partition. Appendix B specifies the annotation model, schema, and core instructions.

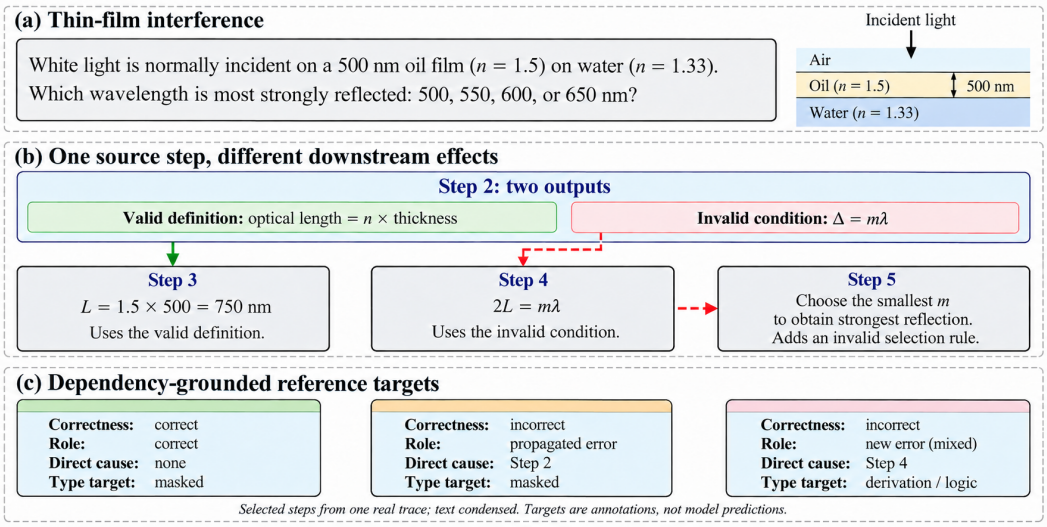


Figure 2: A real thin-film trace links consumed content to supervision. Step s_3 remains correct despite an erroneous predecessor; s_4 inherits an invalid condition; s_5 inherits that condition and introduces a new selection error. Only the intrinsic mistake receives a type target. Green and dashed red arrows show selected valid-content and error-carrying dependencies, respectively.

3.2 Error Introduction and Inheritance

An erroneous predecessor is evidence of propagation only when its invalid content is consumed and materially invalidates the current step. The attribution judgment checks this condition against the problem and original step text. Let A_i contain the directly responsible earlier errors. The accepted error relations satisfy

$$\mathcal{P} = \{(j, i) : j \in A_i\} \subseteq \mathcal{D}^{\text{step}}, \qquad c_j = \mathsf{I} \quad \text{for every } j \in A_i, \tag{4}$$

where $(j, i) \in \mathcal{D}^{\text{step}}$ when step i references an output or assumption of step j. Thus, error attribution must have dependency support, while the presence of a dependency alone does not establish propagation.

For an incorrect step, let $\nu_i \in \{0, 1\}$ denote the annotation judgment that it introduces an intrinsic mistake. Its role is

$$r_i = \begin{cases} \mathsf{new}, & \nu_i = 1, \\ \mathsf{prop}, & \nu_i = 0 \ \wedge \ A_i \neq \varnothing, \end{cases} \qquad (c_i = \mathsf{I}). \tag{5}$$

Here ν_i concerns a local mistake, whereas A_i records inherited causes. A step with both is mixed and receives the new-error target, retaining its inherited causes and supervising its local mistake's type when known. Every propagated error must trace back to a new error through a chain of relations in $\mathcal{P}$. This permits multiple origins within one trace and keeps new-error detection distinct from first-error localization. Figure 2 illustrates how consumed content separates correct reasoning, inherited invalidity, and an additional intrinsic mistake.

Recovery requires a contextually correct step that establishes a justified correction or replacement route. For each recovery event, we mark the earliest such step as $\mathsf{cessation}$ and record the errors it corrects or bypasses in B_i:

$$\mathcal{H} = \{(j, i) : j \in B_i, \ r_i = \mathsf{cessation}\}. \tag{6}$$

A bypass need not consume the erroneous output, so $\mathcal{H}$ is not required to be a subset of $\mathcal{D}^{\text{step}}$. Other correct steps receive $\mathsf{correct}$; insufficient evidence gives $\mathsf{unverifiable}$. Structural checks enforce reference validity and role–cause consistency. Appendix B.3 gives the judgment contract.

3.3 Hierarchical Process Supervision

Dependency attribution separates negative correctness supervision for invalid consequences from intrinsic type supervision for locally introduced mistakes. Let $\mathcal{I}$ contain the steps from accepted,

fully evaluable traces. Correctness targets are $y_i^C = c_i$; every step receives a correct, new, or propagated role target, with cessation merged into correct. Recovery restores contextual validity; its relations remain in the annotations, while the student predicts validity and error origin. For intrinsic types τ_i in the eight-class taxonomy $\mathcal{T}$, the supervised sets are

$$\mathcal{I}_C = \mathcal{I}_R = \mathcal{I}, \qquad \mathcal{I}_T = \{i \in \mathcal{I} : r_i = \textsf{new},\ \tau_i \in \mathcal{T}\}. \tag{7}$$

On $\mathcal{I}_T$, $y_i^T = \tau_i$ describes the mistake introduced at the current step, including mixed errors. Propagated consequences and unknown types contribute no type loss, while retaining correctness and role supervision. Table 8 gives the full mapping. A Qwen3-4B backbone encodes z_i into the target-step representation h_i. Three parallel linear heads predict

$$p_i^k = \mathrm{softmax}(W_k h_i + b_k), \qquad k \in \{C, R, T\}. \tag{8}$$

The hierarchy determines target eligibility; it does not impose a hard prediction cascade. For a batch $\mathcal{B}$ of target-step examples, we use

$$\mathcal{L}_k = -\frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B} \cap \mathcal{I}_k} w_{y_i^k}^k \log p_i^k(y_i^k), \tag{9}$$

$$\mathcal{L} = \mathcal{L}_C + 0.5\mathcal{L}_R + 0.5\mathcal{L}_T, \tag{10}$$

where w^k are class weights and an empty eligible set contributes zero. The denominator counts all input examples, including those masked for a task. Correctness therefore covers propagated errors, while type gradients arise only from known intrinsic mistakes. Appendix C specifies input packing, initialization, weights, and optimization.

4 Experiments

4.1 Data and Comparisons

Evaluation sets. Process verification uses PRM800K (Lightman et al., 2023) (410 problem clusters, 1,377 trajectories, 9,073 steps) and the physics subset of PRMBench Preview with additional annotations (226 problems, 635 trajectories, 3,529 steps). Both use the same Qwen3.8-27B input–output extraction and attribution protocol (Appendix B). ProcessBench (Zheng et al., 2025) evaluates first-error localization and correct-trace acceptance on 3,400 trajectories. Appendix A describes the data sources and separate provider collection.

Existing models and shared-label training. Existing binary, three-class, and hierarchical Qwen3-4B PRMs receive common inputs within each evaluation but differ in training history. On PRM800K, the shared-label study fixes step boundaries, reference labels, initialization, and training settings while varying first-error masking, BFE-style full-trace binary supervision, and role/type objectives. The BFE-style baseline adapts BFE (Yang et al., 2025) to the shared correctness targets (Appendix C). Training/validation groups exclude connections to test by trace identity, problem identity, and normalized text. All shared-label models are selected by validation Error AP. Table 1 compares existing models; Table 2 isolates objective changes under shared initialization and training budget. Appendix C gives the configurations.

4.2 Metrics and Selection Protocols

Error AP is the primary step-ranking metric; error F1, balanced accuracy, role macro-F1, and exact first-error accuracy (FE) measure complementary outcomes. Shared-label thresholds maximize validation balanced accuracy; existing-model evaluations use 0.5. ProcessBench F1 combines localization on erroneous traces with acceptance of correct traces. Score mappings, including its correctness–role decoder, are specified in Appendix D. Reported 95% intervals use 2,000 paired problem-cluster bootstrap resamples and quantify test-sampling uncertainty. Physics, ProcessBench, and downstream results report point estimates. For GSM8K answer selection, methods share 1,319 questions and eight fixed candidates per question; budget N takes the first N candidates. Both PRMs normalize correctness probabilities over the correct/error alternatives and apply identical trajectory aggregation and weighted voting, compared with Majority (Appendix D.3). A secondary study compares PRM-selected SFT with random-subset, full-data, and base conditions (Appendix F.2). Both studies use the application models, separately from the shared-label comparison.

Table 1: Existing-model process verification (%). Decisions use threshold 0.5 with the score mappings in Appendix D. FE includes fully correct trajectories; Role F1 is macro-averaged. Models differ in training history. N/A denotes an unavailable head.

Model	Error AP	Error F1	Bal. acc.	New AP	Role F1	FE
PRM800K: 1,377 trajectories / 9,073 steps						
Binary	72.85	71.68	72.41	N/A	N/A	41.03
Three-class	75.77	72.75	74.51	N/A	N/A	47.35
Hierarchical	**86.33**	**78.99**	**80.78**	62.52	69.92	**55.92**
Annotated physics subset: 635 trajectories / 3,529 steps						
Binary	70.66	64.52	74.32	N/A	N/A	56.38
Three-class	75.83	71.05	79.41	N/A	N/A	61.10
Hierarchical	**84.23**	**73.27**	**79.89**	66.50	68.75	**68.35**

5 Results and Analysis

5.1 RQ1: Error Detection in Mathematics and Physics

Table 1 compares existing models on both annotated evaluation sets. On PRM800K, hierarchical PRM improves Error AP over the three-class model by 10.55 points (95% CI $[8.88, 12.21]$) and by 8.40 points on physics, where the largest advantage is in error ranking. Role macro-F1 is 69.92% on mathematics and 68.75% on physics, providing new-versus-propagated diagnosis in both domains under the common annotation protocol.

Detection gains persist on longer traces. Table 4 shows AP gains across all three length groups. On traces with at least eight steps, AP rises from 74.98% to 84.99%. The long-minus-short error-recall gain is $+4.82$ points (95% CI $[0.88, 8.80]$), although AP gains do not increase with length. Group definitions and input lengths are specified in Appendix D.

5.2 RQ2: Error-Role Diagnosis and Supervision Coverage

Role supervision improves error-origin diagnosis. Table 2(b) compares learned roles with a sequential rule applied to BFE-style binary scores: the first predicted error is new, later predicted errors are propagated, and predicted-correct steps remain correct. The role head raises macro-F1 from 66.24% to 70.80% ($+4.56$ points, 95% CI $[3.37, 5.72]$), with new-error F1 rising from 46.81% to 58.90%. Explicit role supervision thus adds diagnostic information beyond correctness scores and step order.

Diagnostic gains accompany similar detection AP. Hierarchy and BFE-style binary differ by $+0.05$ AP points (CI $[-0.31, 0.43]$), with attribution labels held fixed. Adding role to correctness+type raises conditional type macro-F1 from 20.54% to 26.67%, most strongly for method/formula errors (Table 14). Figure 3 shows student role confusions against reference targets; threshold selection and paired uncertainty are specified in Appendix E.4.

Full-trace coverage supports detection. BFE-style binary raises AP over first-error masking by 12.05 points (CI $[10.79, 13.33]$). After the first error, error recall improves while correct-step recall decreases (Appendix E). This comparison assesses coverage within the shared-label training setup.

5.3 RQ3: Detection versus First-Error Localization

Localization and correct-trace acceptance respond differently. On ProcessBench (Table 3), hierarchy raises mean F1 from 58.61% to 61.84%: gains are largest on OlympiadBench ($+10.80$ points) and Omni-MATH ($+7.49$), while GSM8K loses 7.76 points. Acceptance improves on all four subsets; localization improves only on OlympiadBench and Omni-MATH (Table 18). These outcomes include the effects of the deployed correctness–role decoder.

Table 2: Shared-label detection and diagnosis on PRM800K (%). All models share initialization and training budget; model selection uses validation AP. (a) BFE-style binary uses the full-trace correctness projection of our dependency-derived labels. Decision thresholds are validation-selected; FE (err.) uses erroneous traces. Type F1 averages six supported types on reference new errors with known types. (b) Role prediction; the rule's threshold maximizes validation role macro-F1. Role F1 averages three classes. N/A denotes an unavailable head; bold marks column maxima.

(a) Supervision objectives

Supervision	Error AP	Error F1	Bal. acc.	FE (all)	FE (err.)	Role F1	Type F1
First-error mask	74.48	69.57	72.60	49.09	54.32	N/A	N/A
BFE-style binary	86.53	78.94	80.62	**56.94**	**56.19**	N/A	N/A
Correctness + type	86.54	78.70	80.45	**56.94**	**56.19**	N/A	20.54
Correctness + role	**86.59**	**79.39**	80.88	56.14	55.78	70.72	N/A
Full hierarchy	86.58	79.34	**81.06**	**56.94**	55.15	**70.80**	**26.67**

(b) Role diagnosis from the same binary and hierarchical models

Predictor	Role macro-F1	New F1	Propagated F1
BFE-style binary + sequential rule	66.24	46.81	69.61
Hierarchical role head	**70.80**	**58.90**	**71.81**

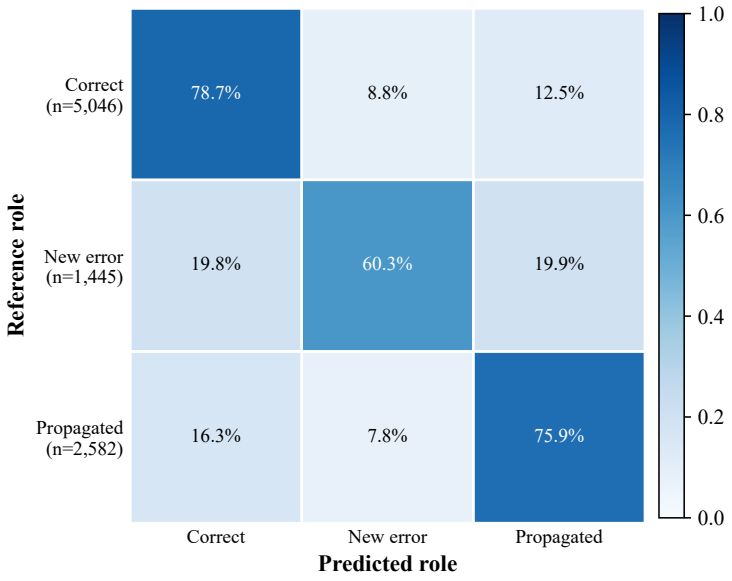


Figure 3: Role predictions of the shared-label full hierarchy (seed 42). Rows normalize the 9,073 test steps by reference role; supports are shown on the left. New errors are more often confused with propagation than the reverse. Reference roles are the augmented annotations; recovery maps to the correct class.

Localization improves for middle and late first errors. On erroneous PRM800K traces, overall FE is similar (54.94% versus 54.11%), but the position breakdown in Table 4 reveals gains of +6.99 and +13.12 points for middle and late first errors, alongside an early-error decline. In the late-error group, correct-step false positives fall from 25.96% to 15.19%, consistent with fewer premature error flags. Overall FE, including correct traces, rises from 47.35% to 55.92%, primarily through fewer false alarms on correct traces. Sparse-error AP declines from 47.56% to 33.67% (Appendix E.3).

Table 3: ProcessBench F1 (%) at threshold 0.5. Average is the unweighted mean over four subsets. Score definitions and the localization/acceptance decomposition appear in Appendices D and E.3.

Model	GSM8K	MATH	OlympiadBench	Omni-MATH	Average
Binary	75.07	55.70	36.08	38.83	51.42
Three-class	**75.81**	64.01	45.84	48.77	58.61
Hierarchical	68.06	**66.41**	**56.64**	**56.26**	**61.84**

Table 4: Existing-model comparison by trajectory length and first-error position on PRM800K (%). n counts trajectories; differences are hierarchical minus three-class, with paired 95% CIs. Position is the reference first-error index divided by trace length; early/middle/late are thirds. Position-group percentages use the 961 erroneous traces; length groups include correct traces. Models are those of Table 1.

Group	n	Three-class	Hierarchical	Difference [95% CI]
Error AP by trajectory length				
≤ 6 steps	839	75.75	85.67	$+9.92\ [7.96, 11.78]$
7 steps	247	77.76	89.84	$+12.09\ [9.00, 15.33]$
≥ 8 steps	291	74.98	84.99	$+10.01\ [6.57, 13.28]$
Exact first-error accuracy by reference error position				
Early (44.64%)	429	69.46	56.64	$-12.82\ [-18.75, -6.53]$
Middle (38.71%)	372	45.97	52.96	$+6.99\ [0.87, 13.35]$
Late (16.65%)	160	36.88	50.00	$+13.12\ [4.48, 21.35]$

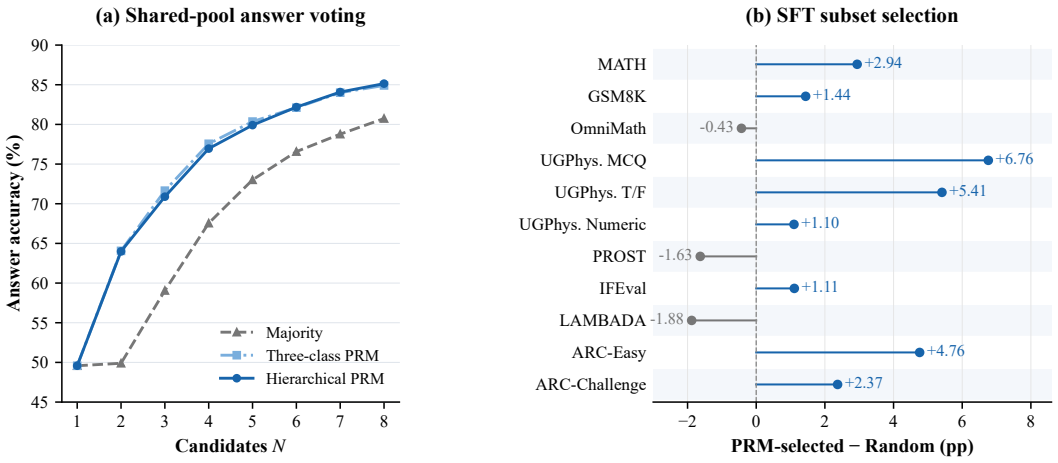


Figure 4: Downstream uses of correctness scores. (a) GSM8K answer selection on 1,319 shared questions, using the first N candidates and the same weighted vote for both PRMs. (b) Post-SFT differences between PRM-selected and equally sized random subsets (67,652 records each at selection), showing all 11 benchmarks reported here. Lines show score differences, not confidence intervals. Full scores appear in Tables 19 and 20.

5.4 RQ4: Downstream Utility of Process Scores

Process weighting improves shared-pool answer voting. Both PRMs outperform Majority at every candidate budget from 2 to 8 (Figure 4(a)). At $N = 8$, hierarchical voting reaches 85.14%, compared with 80.74% for Majority and 84.91% for three-class voting. The 4.40-point gain over Majority supports the utility of process scores beyond answer frequency. The 0.23-point difference between PRMs is a small point estimate without paired uncertainty; roles and types do not enter either voting rule.

Process scores also support training-data selection. Figure 4(b) shows the task-dependent effects of PRM-selected SFT. It reaches 57.66% on MATH versus 54.72% for an equally sized random subset. UGPhysics multiple-choice accuracy rises from 39.86% to 46.62%, while some general capabilities decline relative to the base generator. These single-run results show task-dependent utility of correctness-based selection (Appendix F.2).

6 Discussion

A correctness label records whether a step is valid, but does not distinguish a newly introduced mistake from inherited invalidity or their combination. Input–output attribution separates two questions: whether the step uses an invalid earlier output that undermines its reasoning, and whether its own operation introduces another mistake. The first concerns the particular content consumed, not the correctness of its source step as a whole; the second requires judging the local operation. In Figure 2, s_4 and s_5 both inherit invalidity, but only s_5 introduces an additional mistake. This distinction gives supervision a way to represent both sources of error within one step.

The sequential baseline treats the first predicted error as new and all later predicted errors as propagated. This assumes a single error origin and excludes later independent or mixed errors, even when correctness is predicted accurately. Explicit role supervision provides targets for these cases. With reference labels held fixed, learned roles improve on sequential inference while detection AP remains similar across full-trace objectives (Table 2). Similar correctness rankings can thus accompany different diagnostic capabilities: role evaluation tests whether a model distinguishes the introduction of a mistake from its consequences.

Exact first-error localization requires detecting the mistake while avoiding earlier false alarms. In sparse traces, lower error recall accompanies fewer false positives (Table 16), whereas lower AP indicates weaker ranking across thresholds. The aggregate gains from full-trace supervision therefore coexist with a sparse-error disadvantage under both binary and hierarchical objectives (Table 17). The tested sampling and correctness-weighting variants yield no clear recovery of sparse-group AP. This pattern motivates studying how training coverage and error distributions interact, while evaluating ranking, localization, and diagnosis separately for the intended task.

Attribution depends on the annotation model's judgments about consumed inputs and local operations; systematic errors in these judgments can enter the supervision targets. Extending attribution to implicit premises and unstated domain assumptions calls for further validation of these judgments. Our experiments use a 4B backbone on mathematics and physics, with room to improve role and intrinsic-type prediction and to examine larger models and broader scientific tasks. A concrete next step is to test whether predicted roles help select steps to revise and predicted types help choose corrections.

7 Conclusion

We presented dependency-grounded process supervision for mathematical and physical reasoning. Extracting step inputs and outputs and attributing invalidity through their dependencies yields supervision that teaches a PRM to distinguish newly introduced mistakes from propagated consequences. Evaluations support full-trace supervision for detection and show that the role head improves diagnosis beyond correctness-based sequential inference, alongside downstream selection utility. The work connects error attribution and supervision design, together with evaluations that separate error ranking, diagnostic resolution, and localization.

AI USE STATEMENT

Generative AI tools assisted with manuscript drafting and revision, language and organization, literature discovery, and discussions of research ideas, experimental design, and result interpretation. Generative AI also assisted in creating and editing the framework and case illustrations (Figures 1 and 2). In the research pipeline, Qwen3.8-27B extracted step inputs and outputs and assigned error labels (Appendix B). The authors take full responsibility for the manuscript's claims, results, references, and AI-assisted artifacts.

ETHICS STATEMENT

This study uses existing research datasets and a non-public mathematics and physics collection assembled for this research by a third-party provider. The provider collected and digitized problems from educational materials, including textbooks and postgraduate entrance-examination preparation collections. All collected problems underwent human checking; graduate students engaged by the provider wrote the worked solutions. These solutions may contain reasoning errors and are not treated as uniformly correct.

The provider has confirmed that the supplied collection cannot be publicly redistributed; we therefore do not release it. Sources and uses of the existing research datasets are documented in the experimental sections and appendices.

REPRODUCIBILITY STATEMENT

Section 3 describes the method; Appendices A–D detail data preparation, annotation, training, and evaluation, including the core annotation instructions and output constraints in Appendix B. During the discussion period, we plan to provide an anonymized implementation covering text preparation and annotation validation, label-to-target conversion, training and model selection, and evaluation; experimental configurations, complete annotation prompts, and examples permitted for release will accompany it. Restricted access to the provider-supplied collection limits independent reproduction of experiments using it.

REFERENCES

Aakriti Agrawal, Souradip Chakraborty, Armin Saghafian, Nihal Sharma, Rizal Fathony, Nam H Nguyen, C. Bayan Bruss, Amrit Singh Bedi, and Furong Huang. The hidden bias of process reward models: PRISM for rewarding the right reasoning. *arXiv preprint arXiv:2606.09078*, 2026. URL https://arxiv.org/abs/2606.09078.

Jiarui Che, Yifei Chen, Zhixing Tian, Chenyang Wang, and Ziguang Cheng. Graph-GRPO: Dependency-aware credit assignment for generative e-commerce search relevance. *arXiv preprint arXiv:2605.31003*, 2026. URL https://arxiv.org/abs/2605.31003.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *Nature*, 645:633–638, 2025. doi: 10.1038/s41586-025-09422-z. URL https://doi.org/10.1038/s41586-025-09422-z.

Yuxuan Dong, Xinyu Zhang, Lingling Zhang, Han Lai, Pengyu Li, Bifan Wei, Yaqiang Wu, and Jun Liu. PhysPRM: A generative process reward model with fine-grained diagnosis for physics problem solving. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 9410–9427. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.458. URL https://aclanthology.org/2026.findings-acl.458/.

Wu Fei, Shuxian Liang, Yibo Yang, Yang Lin, Jing Tang, Lei Chen, Xiansheng Hua, and Hao Kong. Self-guided process reward optimization with redefined step-wise advantage for process reinforcement learning. *arXiv preprint arXiv:2507.01551*, 2025. URL https://arxiv.org/abs/2507.01551.

Markus Frohmann, Igor Sterner, Ivan Vulić, Benjamin Minixhofer, and Markus Schedl. Segment any text: A universal approach for robust, efficient and adaptable sentence segmentation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 11908–11941. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.665. URL https://aclanthology.org/2024.emnlp-main.665/.

Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. Process reward models that think. *arXiv preprint arXiv:2504.16828*, 2025. URL https://arxiv.org/abs/2504.16828.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023. URL https://arxiv.org/abs/2305.20050.

Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. Deductive verification of chain-of-thought reasoning. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2306.03872.

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. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-emnlp.807. URL https://aclanthology.org/2025.findings-emnlp.807/.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-Math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024. URL https://arxiv.org/abs/2409.12122.

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. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-emnlp.253. URL https://aclanthology.org/2025.findings-emnlp.253/.

Jian Zhang, Bingyi Wang, and Yizhi Liu. CausalOPD: First-wrong-step supervision for distilling causal chain reasoning. *arXiv preprint arXiv:2608.03673*, 2026a. URL https://arxiv.org/abs/2608.03673.

Yao Zhang, Shijie Tang, Zeyu Li, Zhen Han, and Volker Tresp. WebArbiter: A principle-guided reasoning process reward model for web agents. *arXiv preprint arXiv:2601.21872*, 2026b. URL https://arxiv.org/abs/2601.21872.

Zheng Zhang, Ziwei Shan, Kaitao Song, Yexin Li, and Kan Ren. Linking process to outcome: Conditional reward modeling for LLM reasoning. *arXiv preprint arXiv:2509.26578*, 2025. URL https://arxiv.org/abs/2509.26578.

Wanjia Zhao, Qinwei Ma, Jingzhe Shi, Shirley Wu, Jiaqi Han, Yijia Xiao, Si-Yuan Chen, Xiao Luo, Ludwig Schmidt, and James Zou. PRISM-Physics: Causal DAG-based process evaluation for physics reasoning. In *International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=4PZMeopXzP.

Xiangyu Zhao, Hengyuan Zhao, Yiheng Wang, Wanghan Xu, Yuhao Zhou, Qinglong Cao, Zhiwang Zhou, Lei Bai, Wenlong Zhang, and Xiao-Ming Wu. SCI-PRM: A tool-aware process reward model for scientific reasoning verification. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2026b. doi: 10.1145/3770855.3818999. URL https://doi.org/10.1145/3770855.3818999.

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. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.50. URL https://aclanthology.org/2025.acl-long.50/.

Ninghan Zhong, Ahmet Ege Tanriverdi, Kaan Kale, and Sriram Vishwanath. VeryTrace: Verifying reasoning traces through compilable formalism and structured verification. *arXiv preprint arXiv:2606.24124*, 2026. URL https://arxiv.org/abs/2606.24124.

A Data Sources and Evaluation Populations

PRM800K Phase 2 (Lightman et al., 2023) supplies mathematical solutions with both correct and incorrect reasoning, providing error examples for process supervision. Physics evaluation uses the physics subset of `hitsmy/PRMBench_Preview`. For these two evaluations, we construct correctness, role, and intrinsic-type references using the annotation protocol in Appendix B. The physics subset is not the source benchmark's complete official evaluation. ProcessBench (Zheng et al., 2025) contains 400 GSM8K and 1,000 each of MATH, OlympiadBench, and Omni-MATH trajectories.

Table 5: Process-verification reference populations. Role counts are step counts; training and validation splits are specified in Appendix C.

Set	Problems	Traces	Steps	Correct	New	Propagated
PRM800K	410	1,377	9,073	5,046	1,445	2,582
Physics	226	635	3,529	2,439	557	533

The separate provider collection contains problems digitized from textbooks and postgraduate entrance-examination preparation materials. Problems underwent human checking; graduate students engaged by the provider wrote the solutions, which may contain errors. The collection cannot be publicly redistributed.

B Annotation Specification

B.1 Input and Dependency Schema

The input contains the problem, options, solution, and advisory answer claims. Raw fields are retained. The reasoning view normalizes newlines and malformed escaped LaTeX controls, extracts the reasoning region, and removes its trailing final-answer summary. Empty or malformed inputs and answer-only records are excluded; suspected reasoning errors remain eligible for judgment. SaT-3L-SM proposes spans with protected formula, code, and table boundaries. Reconstruction is checked against this normalized view, not the unprocessed source.

Table 6: Compilation output. References must resolve; step dependencies point backward. Span coverage follows Equation 2.

Object	Required content / constraints
Span	Identifier and exact text; spans concatenate to R
Problem entity	Identifier and atomic fact or condition used
Step	Contiguous span identifiers; operation; inputs; assumptions; outputs
Input reference	Source identifier and dependency type
Allowed sources	Problem entity: data/formula/condition; earlier output: result/formula/condition; earlier assumption: condition
Assumption	Newly adopted premise, excluding method choices

B.2 Compilation and Target Construction

Compilation and judgment use Qwen3.8-27B with thinking disabled and temperature zero. Each call returns one JSON object: compilation follows Table 6, and judgment follows the fields and constraints in Appendix B.3. Text preparation is specified in Appendix B.1.

Compilation instruction

Group adjacent spans into reasoning steps, covering each span exactly once in order; never split or rewrite a span. Merge headings and plans with the following operation. Start a new step for a new formula, assumption, target quantity, or independently reused intermediate result. Preserve reasoning errors. Extract only used problem facts as root entities; formulas or premises introduced by the solution belong to steps. Record each step's operation, consumed inputs, explicit assumptions, and atomic outputs, separating content that later steps use independently. Reference the specific problem entities or earlier outputs/assumptions actually consumed; a transitive path alone is insufficient.

Input: problem, solution, advisory answer claims.
Output: fixed steps, dependencies, error/recovery relations, targets.

1. Normalize the reasoning view R and propose candidate spans.
2. Compile steps and dependencies under Equation 2.
3. Judge fixed steps and edit attribution-relevant dependencies (Appendix B.3).
4. Validate reference resolution, temporal order, coverage, and role–cause constraints.
5. Retain fully evaluable traces and construct targets using Table 8.

Algorithm 1: Annotation and target construction

B.3 Attribution Contract

For each fixed step, the judgment returns a role, optional intrinsic type, direct error causes, corrected/bypassed errors, and relevant dependency edits. The earliest new-error index is returned, or null if none exists.

Table 7: Attribution constraints. Causes precede their consumer. Error-cause relations satisfy Equation 4; recovery relations follow Equation 6.

Role	Required fields / constraints
Correct	Empty causes; null type
Cessation	Nonempty corrected/bypassed error set; null type
New error	Intrinsic type or explicitly unknown; optional erroneous direct parents (mixed error)
Propagated error	Nonempty erroneous direct parents; new-error ancestor; null type
Unverifiable	Empty causes; missing-evidence type; excluded from training

Attribution instruction

Use the original problem and fixed step text as primary evidence; compiled dependencies and answer claims are advisory. Preserve boundaries. Judge contextual correctness and assign a role. A valid operation on erroneous input is propagation unless it adds an intrinsic mistake. List only directly responsible earlier errors; edit associated dependencies when needed. Mark cessation at the earliest justified correction or replacement route. An unrelated correct step or copied answer is not recovery. A step that also introduces a new mistake remains incorrect and retains its inherited causes.

Dependency edits are restricted to correctness, error attribution, recovery, and final-answer support; they do not enumerate every possible relation. Reference validity is checked structurally, while erroneous content and intrinsic mistakes require semantic judgment.

C Training Configuration

C.1 Data Split and Model Input

Only fully evaluated traces without unverifiable steps are eligible. Connected groups defined by trace identity, available problem identity, and whitespace-normalized problem text are excluded if they overlap test. A group-hash split (seed 20260921) assigns approximately 5% of the remaining groups to validation: 267,361/13,930/9,073 training/validation/test steps. This excludes exact normalized matches, not semantic near-duplicates or exposure through initialization.

Inputs concatenate the problem, retained prior steps, and target step with double-newline separators and the tokenizer's BOS token when configured. Classification reads the last target-separator token. At 4,096 tokens, retain the most recent contiguous suffix of complete prior steps; if needed, truncate the problem from its end, then retain the tail and terminal separator of an oversized target. No truncation was recorded in the shared-label data.

C.2 Targets and Class Weights

Table 8: Targets for eligible traces. An unknown intrinsic type masks only type loss; recovery and inherited causes remain in the annotation.

Annotation role	y_i^C	y_i^R	Type supervision
Correct / cessation	C	correct	None
New error (including mixed)	I	new	Known intrinsic type
Propagated error	I	prop	None

The eight types are listed in Table 14. Unknown/other types are excluded from $\mathcal{I}_T$ only. Eligibility follows the role: a retained new error may have the missing-evidence type; an unverifiable step is excluded. For training counts n_{kc} and $K_k^+ = |\{c : n_{kc} > 0\}|$,

$$w_c^k = \min\left(4,\ \frac{n_{kc}^{-1/2}}{(K_k^+)^{-1}\sum_{d:n_{kd}>0} n_{kd}^{-1/2}}\right), \qquad w_c^k = 0 \text{ if } n_{kc} = 0. \tag{11}$$

Weights use the training pool before masking; the five main conditions use natural sampling without resampling. Correctness counts (error/correct) are 128,103/139,258; role counts (correct/new/propagated) are 139,258/42,784/85,319. The resulting weights, rounded to five decimals, are

$$w^C = (1.02087,\ 0.97913),$$
$$w^R = (0.73499,\ 1.32601,\ 0.93900),$$
$$w^T = (0.07041,\ 0.05577,\ 0.11272,\ 0.16834,\ 0.16601,\ 0.63689,\ 2.81245,\ 3.97740).$$

Type weights follow the row order in Table 14. The BFE-style binary baseline trains correctness on every step using Table 8's projection, including cessation as correct. It adapts BFE's full-trace supervision to shared references; no independent BFE relabeling is used. The five conditions activate the losses indicated in Table 2, with Equation 10's coefficients for enabled heads. First-error masking retains all steps through the first error and every step in correct traces (151,419 targets); the batch denominator in Equation 9 remains unchanged.

C.3 Initialization, Optimization, and Selection

All shared-label models initialize from the same PRM800K-trained three-class Qwen3-4B PRM. Transfer the backbone and ERROR/CORRECT classifier rows and biases, excluding NEUTRAL. New role/type heads use zero biases and normal weights with the backbone's initialization standard deviation. Each condition is trained once; the seed and shared configuration appear in Table 9.

Table 9: Shared-label training. Each condition completes 16,712 updates; checkpoint selection maximizes validation Error AP.

Setting	Value
Backbone / head sizes	Qwen3-4B; correctness 2, role 3, type 8
Optimizer / learning rate	AdamW; 5×10^{-6}
Schedule / warmup / decay	Cosine; warmup ratio 0.03; weight decay 0.01
Epochs / seed / global batch	2 / 42 / 32
Length / precision	4,096 tokens / BF16
Head dropout / gradient clip	0.1 / 1.0
Attention / memory	Eager attention; gradient checkpointing
Validation interval	Every 500 updates
Devices / accumulation	8 Metax C550; batch 1 per device; accumulation 4
Software	PyTorch 2.8.0; Transformers 4.57.6; Datasets 5.0.1

Table 10: Selected training update and error threshold. Thresholds maximize validation balanced accuracy and are rounded to six decimals.

Supervision	Selected update	Threshold
First-error mask	500	0.126785
BFE-style binary	14,500	0.458106
Correctness + type	8,000	0.474631
Correctness + role	11,500	0.428626
Full hierarchy	11,000	0.469520

Validation trajectories. Figure 5 shows validation AP over training. All conditions complete the same training budget before selection; Table 10 lists the selected updates. BFE-style binary exceeds masking at every recorded validation update.



Figure 5: Recorded validation Error AP; full-trace binary denotes the BFE-style baseline. Dots mark validation-selected checkpoints; the first evaluation is at update 500.

Application-checkpoint selection. The separate application hierarchy was selected by validation balanced accuracy. Its 267,233/14,058 training/validation steps share 1,683 normalized problems, affecting 2,027 of 2,099 validation traces. No normalized-text overlap with test was found in those partitions; initialization exposure remains unaudited.

D Evaluation Protocols

D.1 Score Mappings and Decisions

Let π_E, π_C, π_N be the three-class comparator's softmax probabilities. Binary PRM uses its ERROR probability throughout; the hierarchical mappings are given below.

Table 11: Error scores and decisions. τ is selected on validation; existing-model PRM800K comparisons use $\tau = 0.5$.

Evaluation	Three-class score	Hierarchical score	Error decision
PRM800K	$\pi_E/(\pi_E + \pi_C)$	$p_i^C(\mathsf{I})$	score $\geq \tau$
Physics	π_E	$p_i^C(\mathsf{I})$	score ≥ 0.5
ProcessBench	π_E	a_i	score > 0.5

The ProcessBench decoder uses

$$a_i = p_i^C(\mathsf{I}) \, \frac{p_i^R(\mathsf{new})}{\max\{p_i^R(\mathsf{new}) + p_i^R(\mathsf{prop}), 10^{-8}\}}. \tag{12}$$

It predicts the first step exceeding the threshold, or no error. Type predictions are unused. All process evaluations have a 4,096-token limit and common inputs within each comparison. Physics thresholds are not selected on physics labels; role scores use the independent three-class role head.

D.2 Metrics, Uncertainty, and Diagnostic Slices

Error AP is non-interpolated average precision, $\mathrm{AP} = \sum_j (R_j - R_{j-1}) P_j$, grouping tied scores. Balanced accuracy averages error and correct recall. For one-based steps,

$$k = \min\{i : y_i^C = \mathsf{I}\}, \quad \widehat{k} = \min\{i : \text{error decision at } i\}, \quad \mathrm{FE}(\mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \mathbf{1}[\widehat{k}_t = k_t], \tag{13}$$

with $\min \varnothing = \infty$. PRM800K contains 961 erroneous and 416 correct traces; physics contains 328 and 307. For ProcessBench, $F_{\mathrm{PB}} = 2EC/(E + C)$, where E is FE on erroneous traces and C is no-error accuracy on correct traces; the reported average weights the four subsets equally.

Role macro-F1 averages three argmax classes. New-error AP uses the new-error role probability over all labeled steps. Type metrics condition on reference new errors with known types; unsupported classes contribute zero to the fixed eight-class macro-F1, and are excluded from the six-supported-class average.

Shared-label thresholds maximize validation balanced accuracy; ties prefer the value nearest 0.5, then the smaller value. Paired intervals resample 410 normalized problem clusters 2,000 times, preserving all steps and trajectories and their multiplicity. Metrics pool steps or traces across sampled clusters. Percentile 95% intervals hold checkpoints and thresholds fixed; seeds are 20260923 (shared-label) and 20260922 (existing-model diagnostics). Subgroup comparisons have no multiplicity adjustment.

Length cutoffs are training-set terciles: at most 6, exactly 7, and at least 8 steps, retaining ties. Error density is $\delta = n^{-1} \sum_i \mathbf{1}[y_i^C = \mathsf{I}]$, with groups 0, $(0, 0.25]$, $(0.25, 0.5]$, and $(0.5, 1]$. Density is a reference-label diagnostic; groups may share problems. All PRM800K test inputs are untruncated. For erroneous traces, first-error position is k/n, with the one-based index k and trace length n, partitioned into early, middle, and late thirds. Positions and error densities use reference labels only for post hoc analysis.

D.3 Shared-Pool Weighted Answer Voting

Both models normalize over the correct/error alternatives: three-class PRM uses $q_i = \pi_C/(\pi_C + \pi_E)$, and hierarchical PRM uses $q_i = p_i^C(\mathsf{C})/(p_i^C(\mathsf{C}) + p_i^C(\mathsf{I})) = p_i^C(\mathsf{C})$ because its correctness head is binary. No role/type fusion is used. For m steps,

$$a = \min_i q_i, \quad g = \exp\left(\frac{1}{m}\sum_i \log\max(q_i, 10^{-12})\right), \quad s = 0.5a + 0.5g. \tag{14}$$

Scoring uses BF16 with a batch size of 16 and a 4,096-token limit. Empty trajectories receive zero. Only the first N candidates in generation order are eligible. For M parseable candidates,

$$u_j = \frac{\exp((s_j - s_{\max})/T)}{\sum_{k=1}^{M}\exp((s_k - s_{\max})/T)}, \qquad w_j = \frac{1-\beta}{M} + \beta u_j, \quad \beta = 0.5,\ T = 0.25. \tag{15}$$

Sum weights by normalized answer and select the largest-mass group. Ties prefer the earliest candidate. The group's highest-scoring candidate represents its answer; score ties again prefer the earliest. With no parseable answer, select the highest-scoring candidate. Majority counts normalized answers and uses the same tie rule.

Answer extraction checks, in order, an explicit #### answer, a boxed answer, final-answer phrases, and the last numeric token. Normalize numeric strings before grouping. Split steps on double newlines. History packing considers prior steps from newest to oldest, skipping those that do not fit while still considering older steps; this differs from the contiguous-suffix process-verification input.

E Additional Process Results

E.1 Supervision Coverage

Table 12: Shared-label paired differences (percentage points, 95% CI). Binary denotes the BFE-style baseline; FE (all/err.) uses all/erroneous trajectories. Differences use full-precision predictions and may differ from subtraction of rounded entries.

Metric	Hierarchy minus binary	Binary minus first-error
Error AP	$+0.05$ $[-0.31, 0.43]$	$+12.05$ $[10.79, 13.33]$
Error F1	$+0.40$ $[-0.15, 0.94]$	$+9.37$ $[8.12, 10.63]$
Balanced accuracy	$+0.44$ $[-0.08, 0.95]$	$+8.02$ $[6.93, 9.13]$
FE (all)	0.00 $[-1.55, 1.56]$	$+7.84$ $[5.01, 10.75]$
FE (err.)	-1.04 $[-2.88, 0.81]$	$+1.87$ $[-1.93, 5.57]$

Table 13: Reference-first-error slices in the shared-label study (%). Prefix includes the first error and all steps of fully correct trajectories (5,313 steps, 961 errors); suffix contains later steps (3,760 steps, 3,066 errors). Correct recall uses the 694 correct suffix steps; it does not measure an explicit recovery label.

Supervision	Prefix AP	Suffix AP	Suffix error recall	Suffix correct recall
First-error mask	64.70	89.44	64.55	60.09
BFE-style binary	59.19	93.71	86.17	47.26
Full hierarchy	58.39	93.84	86.59	49.57

E.2 Conditional Type Predictions

Table 14: Conditional type F1 under matched supervision (%). C+T denotes correctness+type; Full adds role supervision. Both use the same 1,444 reference new-error steps with known types; one additional new error has its type masked. N/A denotes no class-wise test support.

Type	Train	Test	C+T F1	Full F1
Calculation	12,763	473	50.38	52.57
Derivation / logic	20,342	707	62.58	65.00
Method / formula	4,980	134	10.26	36.96
Problem understanding	2,233	61	0.00	5.48
Assumption / condition	2,296	56	0.00	0.00
Answer mismatch	156	13	0.00	0.00
Missing evidence	8	0	N/A	N/A
Unit	4	0	N/A	N/A
Macro-F1 (six supported types)			20.54	26.67
Macro-F1 (eight types; unsupported = 0)			15.40	20.00

Adding role supervision most improves method/formula recognition (Table 14). Adding type to correctness+role changes Error AP by less than 0.01 points (95% CI $[-0.11, 0.09]$).

Under natural sampling, removing only correctness class weights changes full-hierarchy AP from 86.58% to 86.53%, role macro-F1 from 70.80% to 70.74%, and supported-type macro-F1 from 26.67% to 26.77%. Role/type class weights and task-loss coefficients remain fixed.

E.3 Detection and Localization Diagnostics

Table 15: Existing-model Error AP by reference error density (%). Counts are trajectories/steps; differences are hierarchical minus three-class, with paired 95% CIs. Length results appear in Table 4. Density strata change error prevalence; AP is undefined for the 416 fully correct traces (2,590 steps).

Stratum	Count	Three-class	Hier.	Difference [95% CI]
Density $(0, 0.25]$	70/526	47.56	33.67	-13.88 $[-22.50, -3.28]$
Density $(0.25, 0.5]$	271/1,831	79.27	81.30	$+2.03$ $[-0.86, 4.87]$
Density $(0.5, 1]$	620/4,126	92.97	96.59	$+3.63$ $[2.85, 4.42]$

The long-minus-short AP gain is $+0.09$ points (95% CI $[-3.76, 3.98]$). The corresponding error-recall interaction is $+4.82$ points ($[0.88, 8.80]$), while the false-positive reduction is smaller in the long group; the balanced-accuracy interaction is -0.51 ($[-2.95, 2.15]$). The sparse-error slice contains 85 erroneous steps out of 526 (16.16%); all 70 traces contain errors. Prevalence contextualizes AP but is identical for both models and does not explain their difference. Table 16 separates its ranking and decision outcomes: lower AP and error recall coexist with fewer false positives. Exact first-error accuracy increases by 11.43 points (CI $[-3.18, 25.81]$); this measures localization within erroneous traces. AP and these decisions use correctness scores (Appendix D).

Table 16: Sparse-error decisions at threshold 0.5 (%). All 70 trajectories contain errors: 85 erroneous and 441 correct steps in total. FE requires the exact first error; its paired difference interval is reported above.

Model	Error recall	Correct-step FPR	Exact FE
Three-class	85.88	40.82	28.57
Hierarchical	77.65	36.96	40.00

Table 17: Post hoc Error AP by reference error density (%) for the shared-label models. Columns use the same strata as Table 15. Upper rows use Table 2's configuration; lower rows vary sampling and correctness weighting for the full hierarchy. Auxiliary-head weights remain fixed.

Training condition	$(0, 0.25]$	$(0.25, 0.5]$	$(0.5, 1]$
First-error mask	49.25	71.54	90.61
BFE-style binary	35.02	82.69	96.37
Correctness + type	35.57	82.76	96.38
Correctness + role	36.00	82.53	96.52
Full hierarchy	36.13	82.63	96.52
Full hierarchy: sampling and correctness weighting			
Natural, no correctness weighting	36.39	82.62	96.50
70% error-enriched, no correctness weighting	35.69	81.88	96.37
Natural, correctness weighted (main setting)	36.13	82.63	96.52

In the sparse group, BFE-style binary minus masking is -14.23 AP points (95% CI $[-23.09, -5.37]$); hierarchy minus BFE-style is $+1.11$ ($[-1.58, 3.87]$). Thus, the trade-off already appears under full-trace binary training and is not specific to role/type objectives. Error enrichment samples training steps with replacement to target 70% errors, retaining the main optimizer settings and update count. Relative to natural sampling without correctness weighting, full-hierarchy error enrichment changes sparse AP by -0.70 points ($[-2.49, 1.01]$), and correctness weighting by -0.27 ($[-1.06, 0.53]$). These tested adjustments show no clear sparse-AP improvement. The exploratory intervals use 2,000 paired bootstrap resamples of the 61 sparse-group problem clusters (seed 20260926), with fixed checkpoints and no multiplicity adjustment.

On PRM800K, the hierarchical-minus-three-class difference is -0.83 points ($[-4.50, 3.16]$) for exact first-error localization on erroneous traces, versus $+30.29$ ($[25.41, 35.02]$) for accepting fully correct traces.

Table 18: Localization and correct-trace acceptance for existing models (%). Each pair is three-class/hierarchical. Error accuracy requires the exact first error; correct accuracy requires accepting the complete trace. Score mappings follow Appendix D.

Evaluation set	Error accuracy	Correct accuracy
PRM800K	54.94 / 54.11	29.81 / 60.10
ProcessBench: GSM8K	67.15 / 54.11	87.05 / 91.71
ProcessBench: MATH	62.12 / 57.91	66.01 / 77.83
ProcessBench: OlympiadBench	52.95 / 54.46	40.41 / 59.00
ProcessBench: Omni-MATH	53.49 / 54.55	44.81 / 58.09

For fixed correctness scores, the joint score in Appendix D satisfies $a_i \leq p_i^C(\mathsf{I})$, so at the same threshold it flags a subset of the steps flagged by correctness alone. This can suppress false alarms or delay detection. The GSM8K loss is a limitation of the evaluated system, including its decoder; this comparison does not establish whether correctness-only decoding would recover it.

E.4 Correctness-Derived Role Baseline

Table 2(b) uses the saved shared-label predictions. For BFE-style binary scores, let $h_i = \mathbf{1}[p_i^C(\mathrm{l}) \geq \tau]$. The sequential rule predicts correct when $h_i = 0$, new when $h_i = 1$ and $\sum_{j<i} h_j = 0$, and propagated otherwise. Choose τ on the validation set by three-class role macro-F1 over $\{0, 0.001, \ldots, 1\}$; ties prefer the value nearest 0.5, then the smaller value. The selected threshold is 0.511. Hierarchical role predictions use argmax; test labels do not select the threshold.

The role-comparison interval in Section 5.2 uses 2,000 paired bootstrap resamples of the 410 problem clusters, with seed 20260925 and fixed checkpoints and threshold. Fixed threshold 0.5 and the original validation-balanced-accuracy threshold give rule macro-F1 of 66.13% and 66.16%. This evaluates role prediction under shared reference labels; annotation-method comparisons remain a separate question.

Figure 3 shows that new errors are confused with both correct and propagated steps at similar rates. Training role counts and class weights are given in Appendix C.

F Downstream Selection

F.1 Shared-Pool Answer Selection

Table 19 gives the values plotted in Figure 4(a), using the scoring and parsing rules in Appendix D.3.

Table 19: GSM8K answer accuracy (%) using the same candidate pool. Both PRMs apply the same weighted vote ($\beta = 0.5$, $T = 0.25$). Budget N takes the first N candidates; bold compares PRM point estimates.

N	Majority	Three-class PRM	Hierarchical PRM
1	49.58	49.58	49.58
2	49.89	**64.06**	63.99
3	59.06	**71.65**	70.89
4	67.55	**77.56**	76.95
5	73.01	**80.36**	79.91
6	76.57	82.11	**82.18**
7	78.77	84.00	**84.08**
8	80.74	84.91	**85.14**

F.2 SFT Data Selection

Hierarchical PRM scores a pool of 480,350 Quizlet mathematics solutions. The selected and random subsets each contain 67,652 records before the common validation split (14.08% of the pool). Selection requires a minimum step-correctness score of 0.5 and a mixed trajectory score of 0.6.

Table 20: SFT results (%). Base receives no additional SFT; random and PRM-selected use equally sized subsets. Results are individual runs; subsets match record counts. Bold marks the highest score per row.

Benchmark	Base	Random	PRM-selected	Full data
MATH	54.04	54.72	**57.66**	53.56
GSM8K	**84.84**	83.32	84.76	83.02
OmniMath	28.50	**29.56**	29.13	27.35
UGPhysics MCQ	43.24	39.86	**46.62**	36.49
UGPhysics True/False	45.95	50.00	**55.41**	51.35
UGPhysics Numeric	0.95	0.80	**1.90**	**1.90**
PROST	**71.33**	68.26	66.63	59.57
IFEval	**35.86**	31.05	32.16	28.65
LAMBADA	**68.54**	58.51	56.63	62.90
ARC-Easy	61.20	60.67	**65.43**	62.08
ARC-Challenge	42.03	43.73	**46.10**	43.39

Table 21: Full-data SFT configuration. The common validation set is used during training. Selected and random subsets match record counts; their effective training-token budgets were not verified.

Component	Setting
Backbone	Qwen3-4B-base
Framework	TRL SFTTrainer with Accelerate
Optimizer	Adafactor; learning rate 5×10^{-6}; weight decay 0.01
Schedule	Cosine decay; warmup ratio 0.03; one epoch
Input / precision	Maximum 4,096 tokens; BF16
Batch	8 devices; batch 1 per device; accumulation 4; global batch 32
Validation	Common 5,000-record split; evaluate and save every 200 updates