DUET-PRM: FACTORIZING PLAN RELIABILITY AND STEP CORRECTNESS IN PROCESS REWARD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Process reward models (PRMs) provide fine-grained supervision for multi-step reasoning, but most existing approaches collapse different sources of reasoning failure into a single step-level reward. We identify this issue as *plan-step conflation*: an incorrect trajectory may arise either from an unreliable plan or from incorrect execution of an otherwise reasonable plan. To address this, we propose DUET-PRM, a generative PRM that separately predicts plan-level reliability and step-level correctness. We construct the required factorized supervision through a fully automated pipeline based on multiple independent executions, without human annotation or auxiliary LLMs. Across four mathematical reasoning benchmarks, the two reward signals provide complementary information, and their combination consistently improves chain discrimination and downstream Best-of-N selection over either component alone. DUET-PRM also achieves the strongest average candidate-selection performance among the evaluated PRMs. Further analysis shows that higher predicted plan scores consistently correspond to higher reasoning success rates, and that plan-level signals can correct selection errors when step-level evidence alone favors an incorrect answer.

1 INTRODUCTION

Large language models (LLMs) have demonstrated remarkable reasoning capabilities, particularly when equipped with tool use and code execution as part of their solution process (Gou et al., 2024; Yang et al., 2024). In this tool-integrated reasoning paradigm, a policy model solves problems by producing a sequence of steps, each combining a natural-language rationale with executable code and its resulting output. This structured, multi-step format makes tool-integrated reasoning a natural fit for test-time scaling, where allocating additional inference-time compute across candidate solutions has been shown to substantially improve accuracy (Snell et al., 2025; Muennighoff et al., 2025; Guo et al., 2025). Realizing this potential, however, requires a reliable signal for distinguishing promising solution paths from failing ones. Process reward models (PRMs) address this need by assigning step-level scores to each reasoning step, providing fine-grained feedback that guides search, candidate selection, and training (Lightman et al., 2024; Wang et al., 2024; Khalifa et al., 2025).

Existing PRMs assign a single score to each reasoning step, treating the quality of a solution as a monolithic judgment (Wang et al., 2024; Zhang et al., 2025; Luo et al., 2024). In practice, however, solution quality can be analyzed along two conceptually distinct dimensions, as illustrated in Figure 1. The first is the plan level: regardless of how faithfully any individual step carries it out, some plans rarely lead to correct answers. The second is the step level: even given a sound plan, a specific step may deviate from its declared stage or execute it incorrectly. This creates what we term the *plan-step conflation* problem: because a single score cannot distinguish between a flawed strategy and a flawed execution, a low score at any step leaves both the source of failure and the appropriate corrective action ambiguous. A search algorithm that selects among candidate chains cannot tell whether to abandon the plan or to fix the step; a training signal built on such scores propagates this ambiguity into the model. Resolving where a chain went wrong therefore requires a reward signal that treats these two sources of failure as what they are: distinct, independently diagnosable components of solution quality.

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

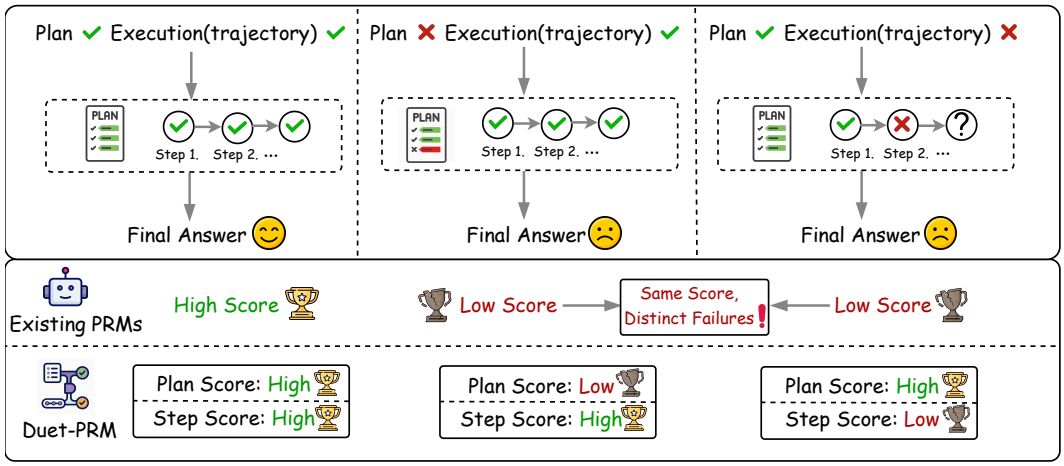


Figure 1: Illustration of *plan-step conflation* in process reward models. A low unified reward may arise either from an unreliable plan or from incorrect execution of an otherwise sound plan. DUET-PRM separates these two cases by predicting plan reliability and step correctness individually.

To address the *plan-step conflation* problem, we propose DUET-PRM, which predicts two scores jointly at each step: a plan score reflecting the reliability of the overall strategy estimated from its behavior across multiple independent executions, and a step score reflecting whether the current step faithfully and correctly executes the declared plan stage. By supervising and predicting these two scores separately, our model can localize the source of failure in a chain without conflating strategy-level and execution-level signals. To train the model without human annotation or auxiliary LLMs, we construct step-level and plan-level labels through a fully automated rule-based pipeline: plan labels are derived from the empirical success rate of a plan across multiple independent executions, and step labels are determined by cross-execution corroboration and plan-stage faithfulness checking. At inference time, the two scores are aggregated within a chain and combined through a weighted sum to produce a single final score, while remaining separately interpretable for diagnosis.

We evaluate DUET-PRM on four mathematical reasoning benchmarks, MATH500, GSM8K, OlympiadBench, and Gaokao2023EN, measuring its ability to discriminate between correct and incorrect reasoning chains via AUC and Cohen's d. Our contributions are as follows:

- We identify the *plan-step conflation* problem in existing process reward models, where strategy-level and execution-level failures are entangled in a single reward signal. We formulate plan-step reward factorization as a principled approach to separately diagnose these two sources of failure in tool-integrated reasoning.

- We develop a fully automated framework for constructing plan-step factorized supervision from multiple independent executions of the same plan. The framework derives plan-level reliability labels from empirical execution success, and step-level correctness labels from cross-execution corroboration and plan-stage faithfulness, without human annotation or auxiliary LLMs.

- Using the resulting supervision, we train DUET-PRM, a generative process reward model that jointly predicts plan-level and step-level scores. Across four mathematical reasoning benchmarks, the two scores provide complementary signals, and their combination consistently improves chain discrimination and Best-of-N selection over either score alone.

2 Related Work

2.1 Process Reward Models

Process reward models (PRMs) provide fine-grained feedback on intermediate reasoning steps and have become an important approach for improving multi-step reasoning beyond outcome-based evaluation (Lightman et al., 2024; Wang et al., 2024; Zheng et al., 2025; Setlur et al., 2025; Zhu et al., 2025). Early work established process supervision as an effective complement to outcome supervision, while subsequent studies have explored different modeling paradigms for process-level evaluation, including scalar, generative, and reasoning-based reward models (Zhang et al., 2025; Khalifa et al., 2025; She et al., 2025; Zhao et al., 2026; Zou et al., 2026). These developments have broadened the ways PRMs can be trained and applied, while highlighting the importance of reliable and informative process-level supervision.

2.2 Automated Process Supervision

Obtaining reliable step-level annotations is substantially more expensive than evaluating final outcomes, motivating methods that construct process supervision automatically or with limited supervision. Early approaches estimate intermediate-step correctness from sampled solutions or final-answer outcomes, while OmegaPRM further automates error localization through Monte Carlo tree search and binary search, enabling large-scale step-level supervision without human annotation (Wang et al., 2024; Luo et al., 2024). Subsequent work has explored supervision signals that avoid explicit ground-truth process labels, including implicit process rewards derived from outcome supervision and automatically generated pseudo-labels (Yuan et al., 2024; Sun et al., 2025; Royer et al., 2026). Other methods enrich process supervision beyond a single forward correctness signal, using bidirectional rewarding signals or explicit error-aware structures to provide more informative feedback (Chen et al., 2025; Pala et al., 2025). More recent work has further investigated unsupervised process reward learning, reducing reliance on both human step annotations and explicit process-level supervision (Gadetsky et al., 2026). These studies explore different ways to reduce the cost of process supervision and improve the informativeness of step-level reward signals.

3 Method

3.1 Problem Setup

We consider mathematical problems solved via tool-integrated reasoning, in which a policy model produces a solution as a sequence of steps, each pairing a natural-language rationale with executable code and its resulting output. Before generating any steps, the policy first produces a plan, a short sequence of labeled stages that outlines the intended solution strategy at a coarse granularity. Each subsequent step declares which plan stage it is executing, giving every trajectory an explicit two-level structure: a plan that organizes the solution into stages, and a sequence of steps that carry out those stages through code execution.

Given such a trajectory, our goal is to train a reward model that produces two scores at each step. The plan score estimates a plan-level reliability property shared across execution prefixes, while the step score reflects whether the current step is faithfully and correctly executed under the declared plan stage. We train both scores as discrete three-way classifications rather than continuous values, a design choice discussed in Section 3.5.

3.2 Plan-Step Factorization

A chain can fail for two reasons that are logically separable but conflated by a single score. The first is a flawed plan: the overall strategy is one that rarely leads to a correct answer, regardless of how faithfully any individual step executes it. The second is a deviated or incorrect execution: the plan itself is sound, but a specific step either departs from the declared stage or carries it out incorrectly. When a single PRM score is assigned to each step, these two failure modes are entangled. A low score could reflect either a bad plan or a bad step, and a downstream consumer of the reward signal, whether a search algorithm or a training objective, cannot determine which. This ambiguity is costly:

diagnosing where a chain went wrong, and therefore how to improve it, requires knowing whether the failure originated in the strategy or in the execution.

We factorize the reward into two components computed jointly at each step. Let a chain consist of a plan P and a sequence of steps $s_1, \ldots, s_n$, each declaring adherence to some stage of P. We define the plan score as an estimate of the shared plan-level reliability of P from the trajectory prefix observed up to the current step, while the step score reflects the correctness of s_i given the preceding history $s_1, \ldots, s_{i-1}$ and the declared stage. At inference time the two scores can be combined, for instance through a weighted sum, to produce a single chain-level or step-level reward, while remaining separately supervised and predicted, enabling us to examine the contribution of the plan score in isolation (Section 4.3).

3.3 Data Generation

We construct training chains by prompting a policy model, Qwen3-8B[1], to solve problems sampled from the MATH training set. The policy first generates a plan for the problem, then produces the sequence of steps that carries it out. We use separate prompts for plan generation and step-level execution; the complete prompts are provided in Appendix A. Throughout the generation process, we use the non-thinking mode. This design is motivated by the self-reflective behavior of thinking-mode models: during execution, the model may detect flaws in a sampled plan and implicitly revise or repair it before generating the corresponding steps. Such behavior would confound plan quality with execution quality, as an initially flawed plan could lead to a successful trajectory after being silently corrected. By using the non-thinking mode, we reduce the tendency of the policy to revise the sampled plan during execution, allowing us to more cleanly disentangle plan quality from execution faithfulness. For each problem, we independently sample M plans, and for each plan we independently sample N chains that carry it out, yielding $M \times N$ candidate chains per problem. We use $M = 4$ and $N = 4$ in our main experiments. Each chain consists of a plan, a sequence of labeled stages drawn from a fixed vocabulary of actions (e.g., `SOLVE`, `COMPUTE`), followed by a sequence of steps, each containing a natural-language rationale, executable Python code, and the resulting output. A chain's final answer is judged correct if it matches the ground-truth answer under a symbolic equivalence check. Applying this procedure to approximately 6,000 problems yields roughly **520K** step-level supervision examples, which we label using the pipeline described in Section 3.4.

3.4 Rule-Based Label Construction

Training the factorized PRM requires step-level and plan-level labels. We construct these labels through a fully automated, rule-based pipeline that uses no auxiliary LLM at any stage, relying instead on structural parsing of the trajectory and cross-execution agreement. Figure 2 illustrates our automated labeling pipeline. Additional implementation details are provided in Appendix D.

Plan-level labels. Given the final verified correctness of each execution, we estimate the plan score as the empirical success rate of the plan across its executions: the fraction of completed executions that reach a verified-correct answer. Plans with a high success rate are labeled *good*, plans with a low success rate are labeled *bad*, and plans with mixed or insufficient evidence are labeled *uncertain*. All prefixes and executions of the same plan share this target, encouraging plan predictions to capture plan reliability rather than execution-specific errors.

Step-level faithfulness. For each step, we check whether its declared plan stage is consistent with the action it actually performs by parsing the code and comparing it against a fixed vocabulary of plan actions. A step is labeled *faithful* if its declared and actual actions match, *deviated* if the policy flags a departure from the plan, and *uncertain* if the action cannot be reliably determined from the code alone. Only faithful steps are eligible to receive a positive correctness label, since we cannot meaningfully verify the correctness of a step that does not do what it claims to do.

Step-level correctness via cross-execution corroboration. For a given plan, we generate multiple independent executions by varying the sampling of the policy. Because faithful steps at the same plan stage should, if correct, converge on the same intermediate result, we treat agreement across at

[1] `https://huggingface.co/Qwen/Qwen3-8B`

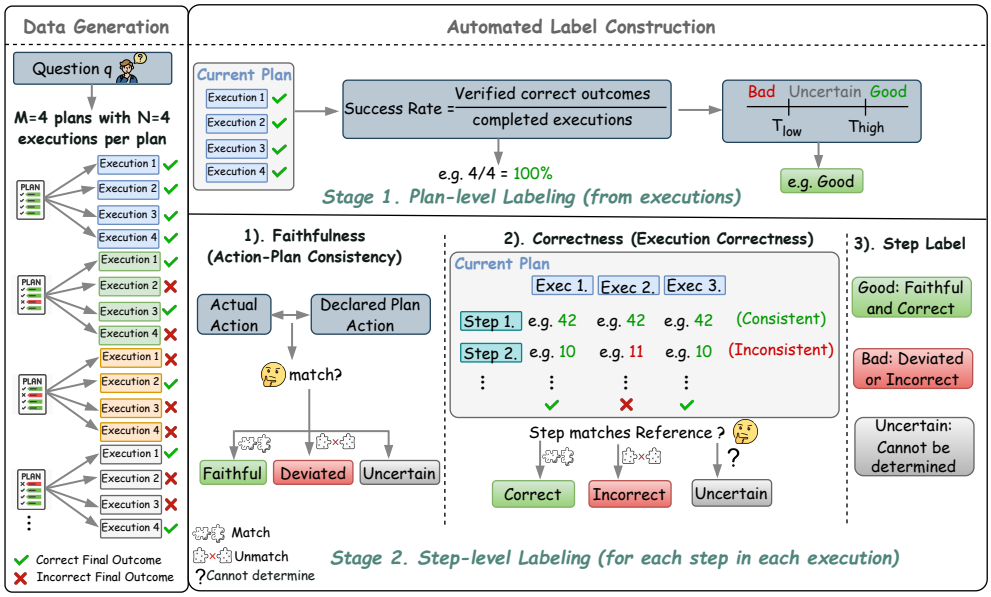


Figure 2: Overview of our fully automated pipeline for constructing plan-step factorized supervision. For each problem, multiple plans and independent executions are sampled. Plan-level supervision is obtained from the empirical success rate of each plan, whereas step-level supervision combines action–plan faithfulness with cross-execution corroboration at corresponding steps. No human annotation or auxiliary LLM is used during label construction.

least two verified-correct executions at a given stage as the reference value for that stage. A step in an incorrect execution is identified as the localized failure point if it is the first step, moving forward through the plan, whose value diverges from this consensus. Steps that agree with the consensus are judged *correct*, while steps that diverge from it are judged *incorrect*. When no reliable consensus can be established, either because too few verified-correct executions reach that stage or because the resulting values are not directly comparable, the correctness judgment is marked as *uncertain* rather than forced into a binary decision. The faithfulness and correctness judgments are then combined to construct the final step-level supervision. A step is labeled *good* if it is faithful and correct, *bad* if it is deviated or incorrect, and *uncertain* if either judgment cannot be reliably determined.

This pipeline produces three categories of supervision for both plan and step scores, *good*, *bad*, and *uncertain*. We retain the uncertain category during training as a valid third class rather than discarding it or forcing it into a binary label. This choice follows directly from the conservative design of our labeling pipeline: at every stage, a judgment is only assigned when the available evidence, cross-execution agreement, action-plan consistency, or final-answer verification, is sufficient to support it. When the evidence is genuinely inconclusive (for instance, too few executions reach a given stage to establish consensus, or two printed values are not directly comparable), forcing a binary label would fabricate a signal where none reliably exists. We treat the uncertain label as an explicit statement that a step or plan cannot be confidently judged from the available evidence, rather than as a placeholder to be resolved later.

3.5 Training Objective

We train the model to output two tags per step, corresponding to the plan score and step score, using standard next-token supervised fine-tuning. Given a trajectory prefix consisting of the problem statement, the plan, and the execution history up to and including the current step, the model is trained to generate a fixed-format response containing exactly two tagged values, `<plan_score>` and `<step_score>`, each taking one of three discrete values, 0.0, 0.5, or 1.0, corresponding to *bad*, *uncertain*, and *good* respectively. Both tags are supervised jointly within a single training example using the standard language modeling loss, with the loss computed only over the tokens comprising

the two score tags rather than the full input context. The reward model is a single generative model that produces both scores through ordinary text generation, in contrast to prior work that attaches a separate classification head to a language model backbone for scalar reward prediction.

At inference time, we parse the two tagged values from the model's output for each step, using the same input format described above, and treat the parsed numeric value directly as the score. When a step is truncated or the model fails to produce a well-formed response containing both tags, we treat the corresponding score as missing rather than imputing a default value, and handle this explicitly during downstream aggregation (Section 3.6).

We fine-tune Qwen2.5-Math-7B-Instruct[2] with full-parameter supervised fine-tuning; we report training details in Appendix B.

3.6 Combining Plan and Step Scores

At inference time, we aggregate the step-level scores within a chain into a single chain-level step score using mean pooling:

$$\text{chain_step_score} = \frac{1}{n} \sum_{i=1}^{n} \text{step_score}_i \tag{1}$$

where n is the number of steps in the chain. The plan itself does not change within a chain, but the model is queried for a plan score prediction at every step, conditioned on the execution history observed so far. We aggregate these per-step plan score predictions into a single chain-level plan score by taking their mean:

$$\text{chain_plan_score} = \frac{1}{n} \sum_{i=1}^{n} \text{plan_score}_i \tag{2}$$

We then combine the chain-level step score and the chain-level plan score into a single final chain score using a weighted combination:

$$\text{final_score} = \alpha \cdot \text{chain_plan_score} + (1 - \alpha) \cdot \text{chain_step_score}, \qquad \alpha \in [0, 1] \tag{3}$$

We select $\alpha = 0.3$ from a set of candidate values on a held-out validation set and fix it across all evaluation benchmarks.

4 Experiments

4.1 Experimental Setup

Policy and PRM backbone. We use Qwen3-4B[3] as the policy model for generating candidate chains in all evaluations. Our PRM is fine-tuned from Qwen2.5-Math-7B-Instruct, the same backbone used by all three PRM baselines. This controls for backbone architecture and model capacity, allowing performance differences to be attributed more directly to differences in reward modeling and supervision. We perform full-parameter supervised fine-tuning using LLaMA-Factory (Zheng et al., 2024); detailed training configurations are provided in Appendix B.

Baselines and benchmarks. We compare against majority voting (without a PRM) and three publicly available PRMs: Qwen2.5-Math-PRM-7B[4] (Qwen2.5-PRM-7B), Skywork-o1-Open-PRM-Qwen-2.5-7B[5] (Skywork-Qwen2.5-7B), and Qwen2.5-Math-7B-PRM800K[6] (PRM800K-7B). We evaluate all methods on four mathematical reasoning benchmarks spanning different levels of difficulty: MATH500 (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021), OlympiadBench (He et al., 2024), and Gaokao2023EN (Zhang et al., 2023).

[2] https://huggingface.co/Qwen/Qwen2.5-Math-7B-Instruct
[3] https://huggingface.co/Qwen/Qwen3-4B
[4] https://huggingface.co/Qwen/Qwen2.5-Math-PRM-7B
[5] https://huggingface.co/Skywork/Skywork-o1-Open-PRM-Qwen-2.5-7B
[6] https://huggingface.co/Qwen/Qwen2.5-Math-7B-PRM800K

Table 1: Chain-level discriminative power (AUC and Cohen's d) across four benchmarks. Cohen's d measures the standardized effect size between correct and incorrect chains' score distributions. AUC 95% confidence intervals are reported in Appendix E.

	MATH500		GSM8K		OlympiadBench		Gaokao2023EN	
Model	AUC	Cohen's d	AUC	Cohen's d	AUC	Cohen's d	AUC	Cohen's d
Qwen2.5-PRM-7B	0.7520	1.0039	**0.8685**	**1.8754**	0.6600	0.5718	0.7456	0.9492
Skywork-Qwen2.5-7B	0.8155	1.1660	0.8597	1.3574	0.7380	0.9137	0.7775	1.0084
PRM800K-7B	0.6414	0.6176	0.8005	1.2969	0.5200	0.2466	0.6644	0.6844
DUET-PRM (Ours)	**0.8502**	**1.4795**	0.8483	1.3534	**0.7777**	**1.1095**	**0.7933**	**1.0772**

Table 2: Best-of-N accuracy ($N = 8$) across four benchmarks. Random is the average single-chain accuracy (pass@1); Oracle is the upper bound achievable by an ideal selector. Rows below Majority report PRM-weighted voting, each denoted as an addition to the majority baseline.

Method	MATH500	GSM8K	OlympiadBench	Gaokao2023EN	Avg
Random (pass@1)	59.1	78.4	31.0	49.5	54.5
Oracle	80.2	90.8	53.8	70.4	73.8
Majority	67.0	83.1	40.2	56.4	61.7
+ Qwen2.5-PRM-7B	69.4	**84.4**	40.2	56.4	62.6
+ Skywork-Qwen2.5-7B	68.6	83.8	40.4	56.7	62.4
+ PRM800K-7B	68.0	82.9	40.2	56.1	61.8
+ DUET-PRM (Ours)	**69.6**	83.8	**42.7**	**57.4**	**63.4**

Metrics. For discriminative power, we report chain-level AUC, the area under the ROC curve for distinguishing chains that reach a verified-correct final answer from those that do not, computed on the final combined score (Equation 3). AUC measures how reliably the PRM ranks correct chains above incorrect ones, and is computed with problem-level cluster bootstrap confidence intervals to account for the fact that chains from the same problem are not independent samples. We additionally report Cohen's d (Cohen, 2013), the standardized mean difference between the final scores of correct and incorrect chains, which measures how well-separated the two score distributions are rather than their relative ranking.

For downstream evaluation, we report Best-of-N accuracy. Given N candidate chains for a problem, we group them by their final answer and compute a score for each answer group by summing the final score (Equation 3) of every chain in that group:

$$\text{group_score}(a) = \sum_{c\,:\,\text{answer}(c)=a} \text{final_score}(c) \tag{4}$$

We select the answer with the highest group score, and report the fraction of problems for which this selection matches the ground truth. We compare this PRM-weighted voting scheme against unweighted majority voting, where each chain contributes a vote of 1 regardless of its score.

4.2 MAIN RESULTS

Discriminative power. We evaluate the discriminative power of our PRM by measuring how well its scores separate chains that reach a verified-correct final answer from those that do not. Table 1 reports chain-level AUC and Cohen's d for our method and the three baselines across four benchmarks of increasing difficulty. DUET-PRM achieves the highest AUC on MATH500, OlympiadBench, and Gaokao2023EN, while also obtaining the largest Cohen's d on these three benchmarks. The advantage is particularly pronounced on OlympiadBench, where DUET-PRM reaches an AUC of 0.7777, outperforming Qwen2.5-PRM-7B and Skywork-Qwen2.5-7B by absolute AUC margins of 0.1177 and 0.0397, respectively. In contrast, Qwen2.5-PRM-7B remains stronger on GSM8K. This pattern suggests that plan-step factorization may be especially beneficial when candidate trajectories involve greater variation in strategy and execution quality.

Table 3: Ablation of the plan and step reward components. We compare using only the plan score, only the step score, and the full final score that combines both components.

	MATH500			GSM8K			OlympiadBench			Gaokao2023EN		
Method	AUC	Cohen's d	BoN	AUC	Cohen's d	BoN	AUC	Cohen's d	BoN	AUC	Cohen's d	BoN
Only plan score	0.8162	1.3979	67.4	0.7853	1.0891	83.5	0.7376	0.9760	40.4	0.7440	0.9079	56.6
Only step score	0.7760	1.1248	67.8	0.8025	1.0486	83.5	0.7711	1.0286	41.0	0.7564	0.9201	57.2
Full final score	**0.8502**	**1.4795**	**69.6**	**0.8483**	**1.3534**	**83.8**	**0.7777**	**1.1095**	**42.7**	**0.7933**	**1.0772**	**57.4**

Table 4: Ablation of score aggregation and plan-step combination on MATH500. We compare minimum, last-step, and mean aggregation under product and weighted-sum combinations of the plan and step scores.

	Product		Weight ($\alpha = 0.1$)		Weight ($\alpha = 0.3$)		Weight ($\alpha = 0.5$)		Weight ($\alpha = 0.7$)		Weight ($\alpha = 0.9$)	
Method	AUC	Cohen's d	AUC	Cohen's d	AUC	Cohen's d	AUC	Cohen's d	AUC	Cohen's d	AUC	Cohen's d
Min	0.6477	0.6725	0.7227	0.5755	0.7227	0.9043	0.7926	1.2100	0.8060	1.3875	0.8140	1.4272
Last	0.7320	1.0178	0.7588	0.7967	0.7588	1.0584	0.8005	1.3045	0.8070	1.4446	0.8177	1.4484
Mean	0.8162	1.3979	0.8041	1.2729	0.8502	1.4795	0.8364	1.4442	0.8363	1.4554	0.8374	1.4507

Downstream candidate selection. Table 2 reports Best-of-N accuracy ($N = 8$) under the PRM-weighted voting scheme defined above. DUET-PRM achieves the highest average accuracy of 63.4, improving over majority voting by 1.7 points and over the strongest competing PRM by 0.8 points. The gain is most pronounced on OlympiadBench, where DUET-PRM reaches 42.7, outperforming majority voting by 2.5 points and the strongest competing PRM by 2.3 points. DUET-PRM also achieves the best accuracy on MATH500 and Gaokao2023EN. On GSM8K, Qwen2.5-PRM-7B remains the strongest selector, achieving 84.4 compared with 83.8 for DUET-PRM. The larger gains on more challenging benchmarks suggest that explicitly modeling plan-level and step-level quality may become more valuable as reasoning difficulty increases.

4.3 ABLATION STUDIES

Contribution of plan and step scores. To examine whether the two reward components capture complementary information, we compare *Only plan score*, *Only step score*, and the *Full final score* of DUET-PRM. Table 3 reports chain-level AUC, Cohen's d, and Best-of-N accuracy across the four benchmarks.

The *Full final score* consistently achieves the strongest overall performance, obtaining the highest AUC and Best-of-N accuracy on all four benchmarks and the largest Cohen's d in each case. The relative strength of the individual components varies across datasets: *Only plan score* performs better than *Only step score* on MATH500, whereas *Only step score* is stronger on GSM8K, OlympiadBench, and Gaokao2023EN. This variation highlights that plan-level reliability and step-level correctness capture distinct aspects of trajectory quality, with their relative utility depending on the problem distribution. The consistent advantage of the *Full final score* further supports their complementary roles. Appendix G provides additional evidence through an empirical plan-step taxonomy and conditional evaluation controlling for step-level evidence.

Score aggregation and combination. We further examine the effects of score-combination strategies and chain-level aggregation methods. For this ablation, we conduct experiments on MATH500. Table 4 compares product and weighted combinations of the plan and step scores, together with minimum, last-step, and mean aggregation for computing the chain-level plan score.

Mean aggregation yields the strongest performance across all score-combination settings, suggesting that averaging plan-score predictions across execution prefixes provides a more robust estimate of the shared plan-level reliability than relying on any single prefix. In particular, minimum aggregation can be overly sensitive to an isolated low-confidence prediction, while the last-step score relies only on the final prefix and discards complementary estimates obtained from earlier prefixes. Weighted-sum combinations further outperform the product formulation under the strongest aggregation setting, which supports treating the two reward components as complementary signals rather than mutually constraining factors. The fixed setting $\alpha = 0.3$ achieves an AUC of 0.8502 and Cohen's d of 1.4795, while performance remains broadly stable across alternative weighting settings.

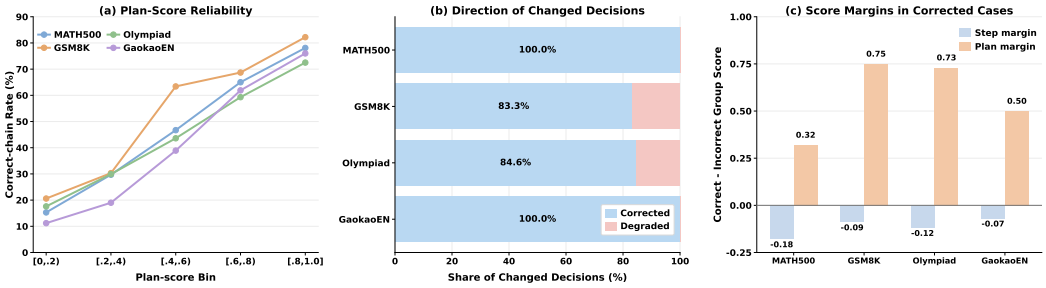


Figure 3: Analysis of the plan-level signal in DUET-PRM. **(a)** Correct-chain rate across plan-score intervals on four benchmarks. **(b)** Direction of selection changes after incorporating the plan score, among problems whose selected answer changes. **(c)** Mean correct-minus-incorrect answer-group score margins in corrected cases under the step and plan signals.

4.4 ANALYSIS AND DISCUSSION

Does the plan score reflect plan reliability? We examine whether the learned plan score preserves the intended notion of plan reliability. As shown in Figure 3 (a), we group trajectories into intervals according to their predicted plan scores and compute the fraction of trajectories in each interval that reach a verified-correct final answer. A clear monotonic pattern emerges across all four benchmarks: trajectories assigned higher plan scores are consistently more likely to produce correct final answers. Correct-chain rates rise from 11.2–20.6% in the lowest-score interval to 72.5–82.2% in the highest across all four benchmarks. This relationship is not confined to the two extreme score ranges. The intermediate intervals also exhibit a progressive increase in correctness across all four benchmarks, indicating that the plan score provides a graded estimate of trajectory reliability rather than merely separating obviously weak and strong trajectories. The consistency of this trend across benchmarks further supports the intended semantics of the plan score as a reliability signal.

When does plan-level information help? We further examine whether plan-level information provides a complementary signal beyond step-level correctness during downstream selection. Figure 3 (b) focuses on problems whose selected answer changes after incorporating the plan score and distinguishes changes that correct an originally wrong decision from those that degrade an originally correct one. Corrected decisions dominate across all four benchmarks, accounting for 100.0% of changed decisions on MATH500 and GaokaoEN, 83.3% on GSM8K, and 84.6% on OlympiadBench. This indicates that, when the plan score changes the final selection, the resulting change is substantially more likely to be beneficial than harmful.

To understand why these corrections occur, Figure 3 (c) compares the score margin between the correct answer group and the incorrectly selected answer group under the step and plan signals. The mean step-score margin is negative on every benchmark, indicating that step-level evidence alone favors the wrong answer in these cases. In contrast, the plan-score margin is consistently positive, showing that the plan signal favors the correct answer group. This complementary signal can therefore reverse decisions that would otherwise remain incorrect under step-level scoring alone.

5 CONCLUSION

We introduced DUET-PRM, a factorized generative process reward model that separates plan-level reliability from step-level correctness to address *plan-step conflation*. We further developed an automated supervision pipeline based on multiple independent executions, without human annotation or auxiliary LLMs. Experiments across four mathematical reasoning benchmarks show that the two signals provide complementary information and that their combination improves chain discrimination and downstream Best-of-N selection. Analysis demonstrates that higher plan scores consistently correspond to higher reasoning success rates and can provide corrective evidence when step-level scoring alone favors an incorrect answer. These results highlight the value of separating planning quality from execution quality when evaluating multi-step reasoning trajectories.

AI USE STATEMENT

We used generative AI tools to assist with language editing, code development and debugging. All research decisions, experimental procedures, experimental results, references, and scientific claims were reviewed and verified by the authors. AI-assisted code was manually inspected and tested before use. We take full responsibility for the final content of this work, including all text, claims, results, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the data generation and automated labeling procedures in Sections 3.3 and 3.4. Appendix A provides the prompt templates used for data generation and reward modeling, Appendix B reports the training configurations and implementation details, Appendix C summarizes the data and label statistics, and Appendix D provides additional details of the automated labeling rules. Appendix E reports the confidence intervals used in the chain-level discrimination evaluation. The experimental setup further specifies the policy and reward-model backbones, benchmarks, baselines, evaluation metrics, and score aggregation procedures.

REFERENCES

Wenxiang Chen, Wei He, Zhiheng Xi, Honglin Guo, Boyang Hong, Jiazheng Zhang, Nijun Li, Tao Gui, Yun Li, Qi Zhang, et al. Better process supervision with bi-directional rewarding signals. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 14471–14485, 2025.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Jacob Cohen. *Statistical power analysis for the behavioral sciences*. routledge, 2013.

Artyom Gadetsky, Maxim Kodryan, Siba Smarak Panigrahi, Hang Guo, and Maria Brbic. Unsupervised process reward models. *arXiv preprint arXiv:2605.10158*, 2026.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Minlie Huang, Nan Duan, Weizhu Chen, et al. Tora: A tool-integrated reasoning agent for mathematical problem solving. In *International Conference on Learning Representations*, volume 2024, pp. 48362–48395, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

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.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison 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*, volume 2024, pp. 39578–39601, 2024.

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

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20286–20332, 2025.

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. *arXiv preprint arXiv:2505.19706*, 2025.

Corentin Royer, Debarun Bhattacharjya, Gaetano Rossiello, Andrea Giovannini, and Mennatallah El-Assady. Process supervision for chain-of-thought reasoning via monte carlo net information gain. *arXiv preprint arXiv:2603.17815*, 2026.

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*, volume 2025, pp. 60808–60838, 2025.

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.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, volume 2025, pp. 10131–10165, 2025.

Lin Sun, Chuang Liu, Xiaofeng Ma, Tao Yang, Weijia Lu, and Ning Wu. Freeprm: Training process reward models without ground truth process labels. *arXiv preprint arXiv:2506.03570*, 2025.

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.

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

Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. Free process rewards without process labels. *arXiv preprint arXiv:2412.01981*, 2024.

Xiaotian Zhang, Chunyang Li, Yi Zong, Zhengyu Ying, Liang He, and Xipeng Qiu. Evaluating the performance of large language models on gaokao benchmark. *arXiv preprint arXiv:2305.12474*, 2023.

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, 2025.

Jian Zhao, Runze Liu, Kaiyan Zhang, Zhimu Zhou, Junqi Gao, Dong Li, Jiafei Lyu, Zhouyi Qian, Biqing Qi, Xiu Li, et al. 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.

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.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 3: system demonstrations)*, pp. 400–410, 2024.

Jiachen Zhu, Congmin Zheng, Jianghao Lin, Kounianhua Du, Ying Wen, Yong Yu, Jun Wang, and Weinan Zhang. Retrieval-augmented process reward model for generalizable mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 8453–8468, 2025.

Jiaru Zou, Ling Yang, Jingwen Gu, Jiahao Qiu, Ke Shen, Jingrui He, and Mengdi Wang. Reasonflux-prm: Trajectory-aware prms for long chain-of-thought reasoning in llms. *Advances in Neural Information Processing Systems*, 38:26764–26802, 2026.

A PROMPT TEMPLATES

Plan Generation Prompt

```
You are a math problem-solving strategist. Read the problem and produce ONLY a solution
↪  plan. Do NOT solve the problem, do NOT compute any numerical values, and do NOT state
↪  any intermediate or final results.

Problem: {problem}

Write a plan of 3-6 ordered stages. Each stage must be:
- a single concrete action that could later be carried out with Python/sympy code;
- verifiable: after execution, one could check whether this stage was done;
- free of any computed values or answers.

For each stage, also give its action type, chosen from EXACTLY this list:
[TRANSLATE, SETUP_EQUATION, SIMPLIFY, SOLVE, CASE_ANALYSIS, ENUMERATE, COMPUTE,
↪  VERIFY_CHECK, FORMAT_ANSWER]

Output ONLY a JSON array, no other text:
[
  {"id": "P1", "desc": "<what to do, no values>", "action": "<one type from the list>"},
  ...
]
```

Step Generation Prompt

```
You are a math problem solver that works strictly by executing a given plan with Python
↪  code.

You will be given a problem and a PLAN with stages P1, P2, .... Solve the problem by
↪  executing the plan stage by stage.

Rules:
1. Work in steps. Each step must follow EXACTLY this format:
<step>
executing: P<k>
thought: <one or two sentences on what this step does for stage P<k>>
```python
<code for this step; use sympy where appropriate; print() clean simplified values only
↪  (numbers/short expressions) | no natural-language sentences, no unsimplified symbolic
↪  dumps>
```
</step>
2. After each step, the system will run your code and return the result in
↪ <output>...</output>. Read it before writing the next step.
3. One stage may take more than one step; declare the same P<k> again.
4. If you must deviate from the plan (skip a stage, reorder, or do something not in the
↪ plan), you MUST declare it:
   executing: DEVIATION (reason: <brief reason>)
5. When the problem is solved, end with:
<answer>...</answer> (final answer only, simplified, containing no natural language)

Problem: {problem}
PLAN: {plan_json}
Begin with your first step.
```

PRM Scoring Prompt

```
You are given a PLAN (a multi-stage solution strategy) and an EXECUTION HISTORY ending in a
↪  CURRENT STEP. Output exactly two scores and nothing else:

plan_score: how reliable is the PLAN itself (independent of how well any single execution
↪  carries it out)? Use exactly one of: 0.0 (this plan rarely leads to a correct answer,
↪  or frequently fails to even produce a complete result), 0.5 (mixed or inconsistent
↪  results across attempts), or 1.0 (this plan reliably leads to a correct answer).

step_score: is the CURRENT STEP itself correct and trustworthy? Use exactly one of: 0.0 (no
↪  | wrong, deviates from the plan, or does no real work), 0.5 (cannot be independently
↪  determined), or 1.0 (yes | well-supported).

Output ONLY the following two tags, in this exact order, with no other text:
<plan_score>X.X</plan_score>
<step_score>X.X</step_score>
```

B TRAINING DETAILS

We initialize DUET-PRM from Qwen2.5-Math-7B-Instruct and perform full-parameter supervised fine-tuning using LLaMA-Factory (Zheng et al., 2024). Training uses DeepSpeed ZeRO-3 and bfloat16 precision, with gradient checkpointing enabled to reduce memory consumption. The maximum sequence length is set to 8,192 tokens.

We train for 2 epochs on 8 NVIDIA A100 40GB GPUs, using a per-device batch size of 1 and gradient accumulation over 32 steps. The learning rate is set to 2×10^{-5} with a cosine learning-rate scheduler and a warmup ratio of 0.03. We reserve 5% of the training data for validation and evaluate every 100 optimization steps. Checkpoints are saved every 100 steps, and the checkpoint with the lowest validation loss is selected as the final model. The corresponding training and validation loss curves are shown in Figure 4.

Table 5: Training configuration of DUET-PRM.

Configuration	Value
Backbone	Qwen2.5-Math-7B-Instruct
Fine-tuning method	Full-parameter SFT
Training framework	LLaMA-Factory
Distributed training	DeepSpeed ZeRO-3
Training hardware	8 × NVIDIA A100 40GB
Maximum sequence length	8,192
Per-device batch size	1
Gradient accumulation steps	32
Learning rate	2×10^{-5}
LR scheduler	Cosine
Warmup ratio	0.03
Training epochs	2
Validation split	5%
Evaluation interval	100 steps
Checkpoint interval	100 steps
Precision	bfloat16
Gradient checkpointing	Enabled
Checkpoint selection	Lowest validation loss

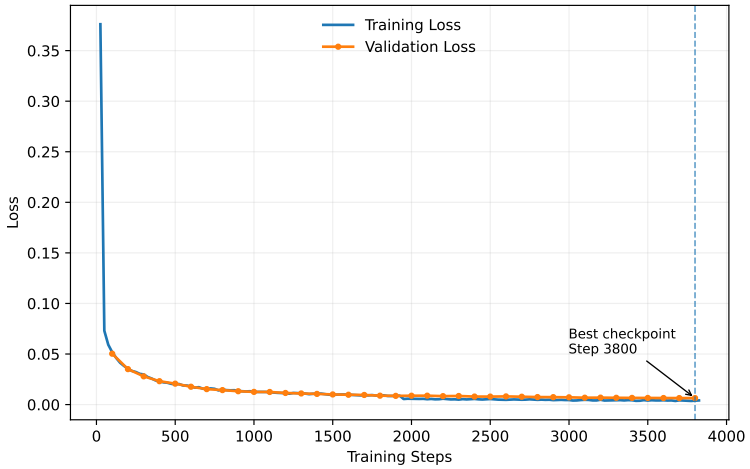


Figure 4: Training and validation loss of DUET-PRM during full-parameter supervised fine-tuning. The dashed vertical line marks the checkpoint with the lowest validation loss.

C Data and Label Statistics

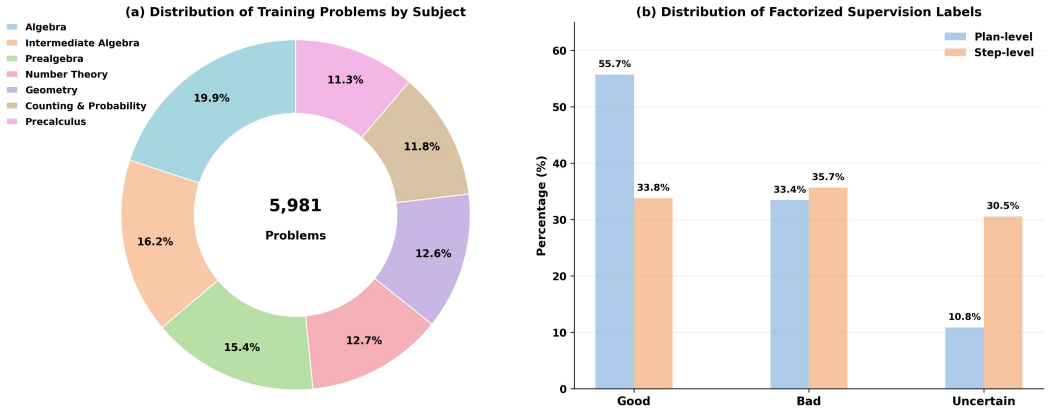


Figure 5: Statistics of the data used to train DUET-PRM. **Left:** subject distribution of the 5,981 mathematical problems used for data generation. **Right:** distribution of the automatically constructed *good*, *bad*, and *uncertain* labels for plan-level and step-level supervision.

D Additional Labeling Details

Plan-label discretization. For each plan, we discretize its empirical execution statistics into three supervision levels. A plan is labeled *bad* if its success rate is at most 0.25, *good* if its success rate is at least 0.75, and *uncertain* otherwise. These thresholds correspond to the natural $1/4$ and $3/4$ boundaries induced by the four independent executions used for each plan. We additionally apply a truncation veto: if at least half of the executions are truncated or incomplete, the plan is labeled *bad* regardless of its success rate among completed executions. This prevents frequently incomplete plans from receiving a high reliability label due to survivor bias.

Rule-based action matching. Step faithfulness is determined using a deterministic AST-based parser over the executed Python code. The parser extracts operation signatures from function calls, arithmetic operations, control flow, comparisons, and variable construction, and maps them to the action vocabulary used in the plan. A step is judged faithful when its observed signature is compatible with its declared action. Cases that cannot be resolved reliably, including malformed syntax, unknown operations, display-only behavior, and ambiguous solve or verification patterns, are left uncertain rather than forced into a binary judgment.

Conservative cross-execution comparison. Cross-execution corroboration is performed only between comparable intermediate values from faithful steps at the same plan stage, while incompatible or inconclusive comparisons are excluded. A failure is localized only when at least two verified-correct executions agree at that stage and an incorrect execution first diverges from this reference. Otherwise, no negative correctness judgment is produced.

E AUC Confidence Intervals

Table 6: 95% confidence intervals for AUC using problem-level cluster bootstrap.

Model	MATH500	GSM8K	OlympiadBench	Gaokao2023EN
Qwen2.5-PRM-7B	[0.7122, 0.8021]	[0.8372, 0.8956]	[0.6180, 0.7007]	[0.6937, 0.7941]
Skywork-Qwen2.5-7B	[0.7764, 0.8564]	[0.8241, 0.8911]	[0.6946, 0.7718]	[0.7181, 0.8131]
PRM800K-7B	[0.5902, 0.6909]	[0.7647, 0.8331]	[0.4755, 0.5631]	[0.6099, 0.7206]
DUET-PRM	[0.8126, 0.8890]	[0.7772, 0.8704]	[0.7532, 0.8224]	[0.7514, 0.8471]

F Qualitative Case Study

Qualitative Example of Plan-Step Factorization

Problem. Suppose $\sqrt{1+\sqrt{2y-3}} = \sqrt{6}$. Find y.

Case A: Reliable Plan, Incorrect Step $\mathbf{P = 1.0,\ S = 0.0}$

Plan. Square the outer radical → isolate the inner radical → square again → solve → verify.

The plan contains the necessary sequence of transformations and provides a valid strategy for solving the problem.

Current step (P2). Starting from

$$\sqrt{2y-3} - 5 = 0,$$

the execution produces only

$$\sqrt{2y-3},$$

instead of preserving

$$\sqrt{2y-3} = 5.$$

The equality constraint is lost, making the current execution step incorrect.

Diagnosis: reliable plan, incorrect execution.

Case B: Unreliable Plan, Correct Step $\mathbf{P = 0.0,\ S = 1.0}$

Plan. Translate the equation → square the outer radical → simplify and solve directly → verify.

The plan omits the explicit isolate-then-square operation required to reliably eliminate the inner radical.

Current step (P1). The execution translates the problem as

```
Eq(sqrt(sqrt(2*y-3)+1),sqrt(6)).
```

This local step faithfully represents the original equation and is supported by the automated step-level supervision.

Diagnosis: unreliable plan, correct local execution.

Takeaway. The same problem admits two distinct failure profiles: a reliable plan can be executed incorrectly, while a locally correct step can occur under an unreliable plan. Factorized supervision distinguishes these cases explicitly.

Figure 6: Qualitative example of plan-step factorization. Green highlights reliable components, while red highlights unreliable or incorrect components.

G Additional Analysis of Plan-Step Complementarity

G.1 Empirical Plan-Step Taxonomy

To further examine whether plan-level and step-level scores capture different aspects of reasoning quality, we group trajectories into four plan-step categories using the neutral threshold of 0.5: high-plan/high-step (HH), high-plan/low-step (HL), low-plan/high-step (LH), and low-plan/low-step (LL). As shown in Table 7, HH trajectories consistently achieve the highest accuracy, while LL trajectories perform substantially worse. The intermediate HL and LH groups further indicate that plan-level and step-level scores capture complementary aspects of reasoning quality.

Table 7: Final-answer accuracy (%) for different plan-step score combinations.

Plan / Step	MATH500	GSM8K	OlympiadBench	Gaokao2023EN
HH	**82.5**	**86.7**	**69.3**	**73.2**
HL	66.0	66.8	40.1	52.6
LH	56.8	57.8	34.3	48.5
LL	24.7	28.5	14.2	20.7

G.2 Conditional Utility of Plan Scores

We further examine whether plan-level scores remain informative after controlling for step-level evidence. For each benchmark, trajectories are partitioned into low, medium, and high step-score regimes according to benchmark-specific tertiles of the chain-level step score. Within each regime, trajectories are further divided into low-plan (LP) and high-plan (HP) groups using the neutral threshold of 0.5, where LP denotes a plan score of at most 0.5 and HP denotes a plan score greater than 0.5.

Table 8: Final-answer accuracy (%) conditioned on step-score and plan-score regimes.

	Step-Low		Step-Mid		Step-High	
Dataset	LP	HP	LP	HP	LP	HP
MATH500	25.9	**69.1**	34.2	**71.5**	60.8	**84.6**
GSM8K	31.2	**69.2**	57.6	**87.7**	63.2	**88.6**
OlympiadBench	15.3	**47.9**	13.5	**30.3**	41.3	**70.4**
Gaokao2023EN	23.1	**62.6**	25.6	**53.2**	45.1	**75.0**

H Limitation

Our current formulation assumes trajectories with an explicit plan structure and identifiable plan stages, which makes the distinction between planning and execution directly observable. Extending the same factorization to less structured reasoning traces, where planning decisions are only implicit in free-form text, may require alternative mechanisms for identifying the corresponding reasoning stages. In addition, the automated supervision pipeline relies on multiple independent executions of the same plan to estimate plan reliability and obtain cross-execution evidence for step-level correctness. Although this construction is performed offline and avoids human annotation or auxiliary LLMs, it introduces additional data-generation cost compared with supervision derived from a single trajectory, motivating more efficient ways of constructing factorized supervision at larger scale.