

SCRIBE: Learning Reusable Skill Rubrics for Process Reward Models

Yuxuan Jiang¹ Francis Ferraro¹

¹University of Maryland, Baltimore County
yuxuanj1@umbc.edu

Abstract

Training reliable tool-augmented agents remains challenging due to the difficulty of **credit assignment** in multi-step reasoning. Process Reward Models (PRMs) provide a natural solution by assigning rewards to intermediate reasoning steps or trajectories, but standard LLM-based judges often produce inconsistent signals due to underspecified evaluation criteria. Recent rubric-based reward methods improve reward quality by making scoring criteria explicit, yet they typically rely on reference answers and construct instance-level rubrics that are difficult to reuse across examples. In this work, we propose **SCRIBE** (Skill-Conditioned Reward with Intermediate Behavioral Evaluation), a reinforcement learning framework for reusable subgoal-level rubric rewards. SCRIBE decomposes reasoning trajectories into intermediate subgoals, clusters semantically similar reasoning behaviors into reusable *Skill Prototypes*, and grounds reward evaluation in prototype-level scoring criteria. This transforms reward modeling from instance-specific open-ended evaluation into reusable process-level verification. Empirical results show that SCRIBE consistently outperforms strong RL and rubric-based reward baselines across mathematical reasoning and tool-use benchmarks; notably, it improves the **AIME25** score of a Qwen3-4B model from **43.3% to 63.3%**. Further analysis shows that SCRIBE reduces dependence on reference answers and provides more stable, cross-instance consistent reward judgments, offering a scalable approach to reliable process reward modeling for tool-augmented agents.

1 Introduction

Tool-integrated reasoning augments the reasoning process with external tool invocations, providing verifiable intermediate signals and improving the performance ceiling compared to traditional text-only reasoning (Yao et al., 2023; Qian et al., 2025). In multi-step and multi-tool settings, errors may

arise from tool selection, argument construction, result interpretation, or intermediate reasoning failures, making outcome-level rewards insufficient for reliable credit assignment (Qian et al., 2025; Zhang et al., 2025). Therefore, Process Reward Modeling (PRM) has emerged as a natural solution by providing denser training signals through rewards assigned to intermediate reasoning steps or full trajectories.

The effectiveness of PRM largely depends on the reliability of reward judgments. However, existing methods often rely on open-ended LLM-as-a-Judge evaluation (Li et al., 2025), where scoring criteria are underspecified. For example, models may receive high rewards despite flawed intermediate reasoning simply because the final answer is correct, while otherwise valid trajectories may be over-penalized due to harmless formatting or syntax errors. To mitigate this issue, recent work introduces explicit rubrics to transform open-ended judging into structured evaluation.

RaR (Rubrics as Rewards) (Gunjal et al., 2025) is a representative rubric-based reward modeling approach that generates instance-level rubrics from the reference answer of each training example, thereby improving the structure and explicitness of reward evaluation. Despite its effectiveness, this approach still has two limitations. First, it depends on high-quality reference answers; we observe that replacing the reference generator from GPT-5.1 to GPT-4.1-nano substantially degrades performance on MATH500, AIME25, and BFCL. Second, instance-level rubrics are difficult to reuse across examples. Each instance requires a separate rubric, increasing construction cost and potentially causing semantically similar reasoning behaviors to be evaluated under different criteria.

Meanwhile, many reasoning patterns repeatedly appear across different tasks, such as constraint verification, boundary handling, tool argument formatting, and retrieval-synthesis alignment. Despite

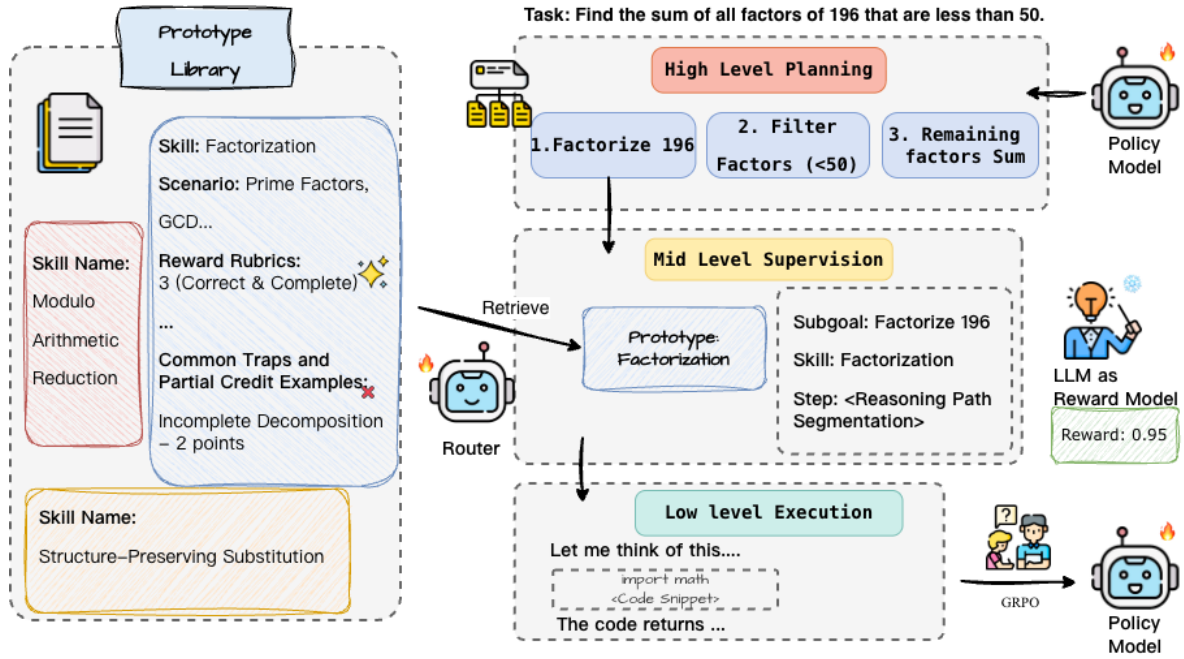


Figure 1: Overview of our three-stage framework. The policy model performs high-level planning, mid-level reasoning, and low-level execution. At the mid level, a router retrieves skill-specific prototypes from a prototype library and uses them to structure reward evaluation by a frozen LLM-based (GPT5-mini) reward model. The policy is optimized with GRPO using the resulting rewards. Flame denotes trainable components, while snowflake indicates frozen models.

this overlap, existing rubric methods still construct independent evaluation criteria for each instance. This motivates a key question: *Can reward modeling reuse structured evaluation criteria across semantically similar reasoning behaviors?*

To address this question, we propose **SCRIBE** (Skill-Conditioned Reward with Intermediate Behavioral Evaluation), a reusable subgoal-level rubric framework for PRM. SCRIBE first decomposes reasoning trajectories into intermediate subgoals and clusters semantically similar reasoning behaviors into reusable skill prototypes. Each prototype captures shared reasoning patterns, common failure modes, and structured scoring criteria that can be reused across different problems.

Rather than generating a new rubric for every instance, SCRIBE transforms reward modeling from instance-specific evaluation into reusable process-level verification. For each intermediate subgoal, a router retrieves the most relevant reusable prototype and provides structured scoring criteria for reward evaluation. By grounding reward assignment in reusable prototype-level rubrics, SCRIBE substantially constrains the open-ended evaluation space of LLM judges, resulting in more stable, consistent, and transferable reward signals.

We evaluate SCRIBE on mathematical reasoning benchmarks (MATH500, AIME25) and tool-use benchmarks (BFCL V4). Experimental results show that SCRIBE consistently outperforms existing RL baselines and instance-level rubric reward methods across different models and domains. Further analysis demonstrates that SCRIBE not only avoids dependence on high-quality reference answers, but also produces substantially more consistent reward judgments across repeated evaluations and semantically similar reasoning behaviors.

In summary, our contributions are as follows:

- This work identifies two key limitations of existing instance-level rubric rewards: dependence on high-quality reference answers and lack of cross-instance reusable evaluation criteria.
- We propose SCRIBE, a reusable subgoal-level rubric framework that transforms reward modeling from instance-specific evaluation into reusable process-level verification through prototype-grounded reward assignment.
- Our work shows that reusable prototype-level rubrics provide more stable and consistent reward signals, improving cross-instance reward consistency and enhancing the robustness and scalability of process reward modeling.

2 Related Works

2.1 Tool-Integrated Reasoning with Agentic Reinforcement Learning

Tool-integrated reasoning extends large language models by enabling them to invoke external tools such as code interpreters and search engines, substantially improving the limits of purely textual reasoning (Yao et al., 2023). Early tool-use frameworks such as Toolformer (Schick et al., 2023) rely primarily on supervised fine-tuning to teach models when and how to invoke external tools. Building on this line, a series of recent *agentic reinforcement learning* approaches, like ToolRL (Qian et al., 2025), DeepSeek-R1 (Guo et al., 2025), explicitly model tool usage as actions and optimize multi-step tool interaction via reward design, such as common use RL paradigm DAPO (Yu et al., 2025) and GRPO (Liu et al., 2025).

Despite their success, most existing methods concentrate on *low-level* tool-use behaviors, focusing on richer invocation patterns, improved execution accuracy, or increased autonomy at the action level (Dong et al., 2025; Xu et al., 2025), rather than evaluating or supervising plan-level discrimination and high-level decision structure. In contrast, it remains underexplored whether tool use improves *mid-level* reasoning quality, such as subgoal execution and intermediate abstraction, or translates into stronger *high-level* reasoning. Our work addresses this gap by examining how agentic tool use affects reasoning competence beyond surface-level tool invocation.

2.2 Progress Reward Modeling with LLM-as-a-Judge

Progress reward modeling (PRM) provides denser supervision by evaluating intermediate reasoning steps rather than only final outcomes (Setlur et al., 2024). Recent work commonly leverages LLM-as-judge frameworks to scale PRM, benefiting from low annotation cost and strong generalization (Li et al., 2025; Choudhury, 2025).

However, LLM-based judges often exhibit inconsistency and vulnerability, which can lead to reward hacking and unstable training (Zhao et al., 2025; Li et al., 2026). These issues arise when models exploit weaknesses in the reward signal rather than improving genuine reasoning quality. Recent work explores rubric-based supervision to improve reliability. RAR (Gunjal et al., 2025) generates instance-level rubrics using golden answers, but

this approach depends on high-quality references and is difficult to scale. In contrast, our method does not rely on golden answers and instead constructs cluster-level reusable rubrics (covering ~ 20 instances per rubric), achieving better scalability and generalization in practice.

2.3 Problem Decomposition and Skills in Reasoning

Among LLMs’ various ability (Yu et al., 2026; Rao et al., 2026), decomposing complex reasoning problems into a sequence of simpler subgoals has been shown to improve performance on challenging tasks (Xiang et al., 2025; Teng et al., 2025; Shi et al., 2022). With an appropriate decomposition, each subgoal typically corresponds to a dominant reasoning skill, where a skill denotes a reusable reasoning operation, such as algebraic factorization or querying external information (Dalal et al., 2024).

Prior work has shown that skills form a stable and model-recognizable level of structure within LLMs (Jiang et al., 2026). Compared to step-level decomposition, skill-based decomposition enables more consistent identification of reusable reasoning patterns across problems. Building on this insight, our work investigates how strengthening tool-related skills can improve subgoal execution and, consequently, overall reasoning performance.

3 Method: Decomposition, Clustering, and Prototype Construction

Our framework, SCRIBE, decomposes the training process into three core stages: skill-level abstraction, structured reward evaluation, and policy optimization. As illustrated in the system overview (Figure 1), we move beyond raw process-level rewards by introducing a mid-level “semantic bridge.” We first distill reasoning trajectories into a library of reusable *Skill Prototypes*. During training, an auxiliary *Router* maps the student’s rollouts to these prototypes, allowing an LLM-based judge to provide rewards that are calibrated against a consistent taxonomic rubric rather than subjective inference. Finally, the model is optimized using GRPO, benefiting from the densified and stabilized credit assignment signals.

3.1 Mid-Level Skill Abstraction

We formalize a trajectory as an ordered sequence of $\langle \text{subgoal}, \text{skill}, \text{step} \rangle$ triples, where each step represents a non-overlapping span of the reasoning trace.

We show a simplified version of prompt here and the full prompt is provided in Appendix D.

Simplified Subgoal and Skill Extraction Prompt

Given a problem or a tool-using trajectory, decompose the solution into a sequence of mid-level subgoals, where completing these subgoals in order is sufficient to solve the whole problem. For each subgoal, identify: (1) what the subgoal accomplishes, (2) the primary reasoning skill required, (3) a representative step or span where the skill is applied. Example: **subgoal:** maximize $u = (x + 4)y$ subject to $x^2 + y^2 = 1$

skill: optimization with constraint (Lagrange multipliers or trigonometric substitution)

step: set $x = \cos \theta$, $y = \sin \theta$, so

$u(\theta) = (\cos \theta + 4) \sin \theta = \frac{1}{2} \sin 2\theta + 4 \sin \theta$,
and maximize over θ .

Prototype Construction. To build this abstraction, we first decompose trajectories into subgoals and associated skills using the prompting scheme in Appendix A. We then apply a hierarchical clustering approach—utilizing HDBSCAN for dense clusters and K-means as a fallback—to group semantically similar reasoning patterns. For each cluster, we distill a representative *Skill Prototype* that aggregates usage contexts, intermediate objectives, and canonical reasoning patterns while abstracting away instance-specific details like concrete numerical values.

Structured Verification via Prototypes. Skill Prototypes transform subjective, open-ended judging into structured verification. As shown in the Bound-Based Conclusion example (Figure 2), once a step is routed to a prototype, the LLM judge is provided with a precise, checklist-style rubric. For instance, instead of vaguely rewarding "logical correctness," the prototype requires verifying if the agent properly addressed inequality strictness or identified necessary bounds. Crucially, each prototype defines Common Traps—such as a "Boundary Leak" (e.g., failing to discretize a continuous bound like $c < 44.75$ to $c \leq 44$)—and maps them to specific penalty scores. This granularity ensures the reward signal is a diagnostic reflection of skill mastery rather than a biased estimate based solely on

Skill Prototype (Compact): Bound-Based Conclusion & Synthesis

Skill: Bound-Based Conclusion and Synthesis. **Use when:** a subgoal *terminates* a reasoning chain by concluding from established bounds or constraints.

Pattern: (i) collect relevant bounds $\rightarrow$ (ii) determine whether tightness/feasibility must be addressed $\rightarrow$ (see Appendix for remaining steps).

Scoring Rubric (summary): **3** rigorous conclusion with tightness; **2** correct conclusion with minor slip; **1** major logical gap; **0** wrong or premature conclusion.

Common Trap (example): *Boundary leak* — failing to discretize a continuous bound (e.g., concluding $c < 44.75$ without stating $c \leq 44$), which leaves the subgoal incomplete (**Score = 2**).

Note: Additional traps (implicit tightness, domain neglect, premature synthesis) and the full scoring protocol are provided in the appendix.

Figure 2: A compact illustration of a skill prototype used for cluster-calibrated judging. Detailed prototypes and trap-to-score mappings are deferred to the appendix.

the final outcome, leading to promising result.

3.2 Skill-Conditioned Reward and Optimization

Routing and Reward Evaluation. To recover mid-level structure during training, we train a lightweight *Router* (Qwen3-4B-Instruct-2507) to map raw trajectories into structured $\langle \text{subgoal}, \text{skill}, \text{step} \rangle$ triples. During RL, the Router assigns student-generated subgoals to their corresponding Skill Prototypes. A judge LLM then evaluates these subgoals with scores in $\{0, \dots, 3\}$ conditioned on the prototype’s checklist. To ensure reward reliability, we adopt a calibration protocol that (i) verifies rewards across multiple prompt variants, (ii) leverages subgoal-level accuracy statistics, and (iii) monitors consistency via repeated evaluation on fixed anchor subgoals. The final process reward is a weighted average of these calibrated subgoal scores. Full details of the routing training and performance are provided in Appendix A.

Policy Optimization with Adaptive Prototypes.

The student model is optimized using the GRPO objective based on these skill-conditioned rewards. To mitigate distributional shifts, we implement an **adaptive refresh** mechanism: every 1,000 training steps, we re-cluster accumulated trajectories to update the Prototype library. This ensures that the supervision remains aligned with the model’s evolving reasoning patterns while maintaining stable abstractions for consistent credit assignment.

4 Experimental Setup

Experimental Settings. We conduct reinforcement learning experiments using Qwen3-4B-Instruct-2507¹ as the primary policy model, and additionally evaluate generality with LLaMA-3.2-3B-Instruct². Process-level rewards are provided by an LLM-based judge (GPT-5-mini)³. The Router is implemented as a lightweight model fine-tuned from the same base model as the policy.

Training is performed on approximately 10k problems sampled from the MATH dataset⁴ and the ToolACE (Liu et al., 2024) dataset. These data are used to construct skill annotations, train the Router, and perform GRPO-based policy optimization. For mathematical reasoning tasks, we primarily rely on the Python interpreter as the tool, which is natively supported by Qwen3 family.

Applying the skill clustering protocol described in Sec. 3.1, we obtain a compact mid-level skill space. Initially, clustering yields 418 skill clusters for mathematical reasoning and 472 clusters for tool-use trajectories. Clusters are periodically refreshed during training to incorporate newly observed trajectories. After convergence, the skill space contains 424 clusters for math reasoning and 503 clusters for BFCL tool-use tasks.

The Router is trained on judge-annotated trajectories with a 9:1 train–test split, achieving 98.6% accuracy in skill prototype retrieval on held-out data. Detailed subgoal-, skill-, and step-level evaluation metrics are reported in Appendix B.

During training, we combine process-level and final-answer rewards, with weights of 0.3 and 0.7

respectively. Unless otherwise specified, this reward composition is used throughout.

The policy model is optimized using GRPO with skill-conditioned rewards. We evaluate mathematical reasoning on MATH500 and AIME25, and assess tool-use generalization on BFCL v4 (Patil et al., 2025) (last updated: 2025-11-03) using the official evaluation scripts. Results are evaluated on latest BFCL V4. As a result, our reported numbers may differ from those reported by prior baselines evaluated on earlier versions of the benchmark. We report pass@1, averaged over eight independent runs to reduce variance. Additional training details and learning curves are provided in Appendix C.

Baselines. We compare against a step-level process reward (PRM) baseline that assigns rewards directly to individual reasoning steps using the same LLM-based judge, without mid-level skill abstraction. We further include strong recent baselines for mathematical reasoning, including ReST (Lin et al., 2025), RaR (Gunjal et al., 2025), and NPR (Wu et al., 2025), following their official released implementations and model checkpoints.

5 Main Result

Table 1 summarizes the main results on mathematical reasoning and general tool-use benchmarks. Overall, SCRIBE consistently outperforms prior approaches across both settings, showing its effectiveness beyond a single task domain.

On **Qwen3-4B-Instruct-2507**, SCRIBE achieves the best performance on MATH benchmarks, improving MATH500 accuracy from 92.3 (PRM) and 90.7 (RaR) to 95.8, and AIME25 accuracy from 53.8 (NPR) to 63.3. On BFCL, SCRIBE also obtains the highest overall score of 51.3, outperforming RaR (46.2) and PRM (44.6), with gains across both single-step and multi-step tool-use scenarios.

We further observe that PRM remains a strong baseline, improving Qwen3 from 89.1 to 92.3 on MATH500 and from 33.0 to 44.6 on BFCL Overall. However, SCRIBE further improves over PRM by +3.5 on MATH500 and +6.7 on BFCL Overall. Compared with RaR, SCRIBE improves by +5.1 on both MATH500 and BFCL Overall, suggesting that reusable skill-level rubrics provide more effective supervision than instance-specific rubric generation during the training process.

A similar trend holds for **LLaMA 3.2-3B-Instruct**, where SCRIBE improves MATH500 ac-

¹<https://huggingface.co/Qwen/Qwen3-4B-Instruct-2507>

²<https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct>

³<https://platform.openai.com/docs/models/gpt-5-mini>

⁴<https://huggingface.co/datasets/open-r1/OpenR1-Math-220k/tree/main>

Method	MATH		BFCL					
	MATH500	AIME25	Overall	Web Search	Memory	Single (Live)	Single (Non-Live)	Multi
Qwen3-4B-Instruct-2507								
Base	89.1	43.3	33.0	5.0	12.7	86.7	74.5	19.4
PRM	92.3	51.7	44.6	10.0	15.2	88.5	77.6	28.6
ReST	88.5	43.3	38.9	5.0	13.1	87.2	76.5	25.3
RaR	90.7	43.3	46.2	15.5	17.9	89.6	79.8	30.4
NPR	91.7	53.8	35.2	5.5	14.2	87.1	73.8	17.7
SCRIBE (Ours)	95.8	63.3	51.3	12.5	15.8	90.0	80.5	33.3
LLaMA 3.2-3B-Instruct								
Base	40.8	1.7	21.5	0.5	5.2	82.0	58.3	3.9
PRM	48.3	6.7	24.8	5.0	8.3	83.8	59.6	8.2
ReST	40.2	1.7	23.2	4.5	9.1	83.2	60.5	5.2
RaR	54.6	6.7	27.0	6.5	9.9	84.2	61.1	9.1
SCRIBE (Ours)	63.4	15.8	30.8	8.5	11.2	85.5	62.0	13.5

Table 1: Main results on mathematical reasoning and tool-use benchmarks. Result are evaluated on latest BFCL V4 (updated on Nov.03 2025, results might be various from origin reports.)

curacy from 54.6 (RaR) and 48.3 (PRM) to 63.4, and BFCL Overall from 27.0 (RaR) and 24.8 (PRM) to 30.8. These results indicate that our prototype-conditioned reward formulation yields more reliable and transferable improvements across reasoning and tool-use tasks.

6 Research Questions and Ablation Studies

6.1 RQ1: Does Reference Dependency Limit Rubric-Based Reward Modeling?

Recent rubric-based reward modeling approaches have been used to mitigate the underspecified evaluation criteria of LLM-as-a-Judge. For example, RaR first generates an instance-level rubric from the reference answer of each training example, and then uses this rubric to score model outputs. This design makes reward criteria more explicit, but it also introduces an important assumption: the quality of the generated rubric depends on the quality of the reference answer. If the reference answer is incomplete, incorrect, or only reflects one valid reasoning path, the resulting rubric may inherit these biases and produce less reliable reward signals.

SCRIBE is designed to reduce this reference dependency. Unlike RaR, which constructs a separate final-answer rubric for each instance, SCRIBE does not use golden answers to extract scoring criteria. Instead, it automatically decomposes model trajectories into intermediate subgoals and clusters semantically similar reasoning behaviors into reusable skill-level prototypes. Each prototype captures shared reasoning patterns, common failure

Dimension	RaR	SCRIBE
Rubric level	Instance-level	Subgoal-level
Reward type	Outcome Reward	Process Reward
Reference answers	Required	Not required
Rubric construction	From golden answers	Decompose + cluster trajectories
Reuse scope	One rubric per instance	One rubric for multiple subgoals
Cross-instance consistency	Limited	Explicitly encouraged
Scalability	Cost grows with instances	Cost grows with prototypes

Table 2: Comparison between instance-level rubric rewards and SCRIBE’s reusable skill-level rubric rewards.

modes, and scoring criteria, allowing the same rubric to be reused across similar subgoals from different instances.

Table 2 summarizes the main differences between RaR and SCRIBE. Overall, RaR provides fine-grained explicit rubrics at the instance level, but its reward construction depends on high-quality reference answers and its rubrics are difficult to reuse across examples. In contrast, SCRIBE shifts rubric rewards from instance-level final rewards to reusable subgoal-level process rewards, providing structured supervision without requiring reference answers, which provides a satisfying result.

Among these differences, the dependence on reference answers is particularly important because it directly affects how reliable the generated rubric can be. To examine the effect of reference quality

Method	Reference	MT500	AIME25	BFCL
Qwen3-4B	–	89.1	43.3	33.0
RaR	GPT-5.1	90.7	43.3	46.2
RaR-Weak	GPT4.1-nano	84.9	31.7	39.5
SCRIBE	None	95.8	63.3	51.3

Table 3: Effect of reference dependency in rubric-based reward modeling. When the reference source is weakened, RaR performance drops substantially, while SCRIBE avoids this dependency by constructing reusable skill-level rubrics without golden reference answers.

on RaR, we conduct a lightweight diagnostic experiment. Specifically, we replace the original strong reference generator (GPT-5.1) used in RaR with a weaker model, GPT-4.1-nano-2025-04-14⁵, to generate reference answers and construct instance-level rubrics. All other GRPO training settings remain unchanged. We denote this setting as RAR-WEAK, based on the model setting.

This replacement provides a direct test of whether RaR remains effective when reference quality decreases. The weaker reference model itself is substantially less capable than GPT-5.1: it achieves 24.88 BFCL overall performance compared with 55.46 for GPT-5.1, and its average performance on MATH500 and AIME25 drops from 62.6 to 34.1. Thus, if RaR performance degrades under this setting, it indicates that reference-dependent rubric construction can become an unstable source of supervision.

As shown in Table 3, replacing GPT-5.1 with GPT-4.1-nano-2025-04-14 substantially reduces RaR performance on MATH500, AIME25, and BFCL. This suggests that the effectiveness of instance-level rubric rewards depends not only on rubric-based scoring itself, but also on the quality of the reference answers used to generate those rubrics. By contrast, SCRIBE does not require reference answers and instead constructs reusable scoring criteria through trajectory decomposition and prototype clustering. Therefore, SCRIBE avoids the error propagation chain from low-quality references to rubrics and reward signals, providing a more stable and scalable process reward for GRPO training usage.

⁵<https://openai.com/index/gpt-4-1/>

Method	Exact Agree $\uparrow$	Reward Std. $\downarrow$
No rubric PRM	71.8	0.63
RaR	82.4	0.41
SCRIBE	93.7	0.18

Table 4: Repeated evaluation consistency under identical inputs. For each sample, reward evaluation is repeated 10 times with independent sampling. SCRIBE produces substantially more stable reward assignments than direct PRM and instance-level rubric rewards.

6.2 RQ2: Does SCRIBE Produce More Consistent Reward Judgments?

Beyond reference dependency, we further investigate whether different reward modeling methods can maintain consistent evaluation standards for similar reasoning behaviors.

Instance-level rubric methods construct a separate rubric for each problem based on its corresponding reference answer. As a result, evaluation criteria may drift across instances, even when two reasoning trajectories contain highly similar intermediate behaviors or failure patterns. In contrast, SCRIBE groups semantically similar subgoals into shared reusable prototypes and applies the same scoring criteria across different instances, which should theoretically encourage more stable reward judgments, comparing to no rubric.

Repeated Evaluation Stability. We first study whether reward assignments remain stable under repeated evaluations with identical inputs. Specifically, for the same: (i) problem, (ii) reasoning trajectory, (iii) rubric, and (iv) judge model, we repeat reward evaluation 10 times with independent sampling. We report two metrics: (1) **Exact Agreement**, the percentage of repeated evaluations producing identical reward scores; and (2) **Reward Std.**, the standard deviation of rewards across repeated evaluations.

As shown in Table 4, SCRIBE achieves the highest repeated-evaluation agreement and the lowest reward variance across all settings. Because SCRIBE grounds evaluation in reusable prototype-level scoring criteria, its evaluation space is substantially more constrained than direct LLM judging or independently generated instance-level rubrics. Therefore, SCRIBE is expected to produce more stable reward assignments. These results suggest that reusable prototype grounding substantially reduces reward fluctuation and provides more stable optimization signals for GRPO.

Method	Exact Agreement $\uparrow$	Avg. Score Gap $\downarrow$
Direct PRM	58.6	0.82
RaR	69.4	0.57
SCRIBE	86.8	0.24

Table 5: Cross-instance reward consistency on 50 manually matched reasoning behavior groups. Each group contains semantically similar subgoals from different problems. SCRIBE produces more consistent judgments across matched reasoning behaviors than direct PRM and instance-level rubric rewards.

Cross-Instance Consistency. We further evaluate whether reward methods assign consistent scores to similar reasoning behaviors across different problems. To this end, we construct a controlled cross-instance evaluation benchmark. Specifically, we manually collect a small subset containing 50 reasoning behavior groups. Each group contains 2 to 11 subgoals, with an average of 5.2 subgoals per group. The subgoals within each group are drawn from different problems but share the same reasoning pattern or failure type, such as boundary leak, missing constraint verification, incorrect API argument formatting, premature conclusion, or correct retrieval but incomplete synthesis.

For each group, we extract only the corresponding subgoal spans rather than the full trajectories. This allows the evaluation to focus on the target reasoning behavior while minimizing interference from unrelated errors. For SCRIBE, we directly apply the corresponding reusable prototype scoring criteria. For RaR, we extract the most relevant scoring dimension from its instance-level rubric and use it to evaluate the target subgoal, ensuring a controlled and fair comparison.

We report two metrics: (1) **Exact Agreement**, the percentage of subgoal pairs within the same group that receive identical reward scores; and (2) **Average Score Gap**, the average absolute reward difference between subgoal pairs in the same group. Lower score gaps and higher agreement indicate stronger cross-instance consistency.

As shown in Table 5, SCRIBE achieves substantially higher agreement and lower score gaps than both Direct PRM and RaR. This indicates that reusable prototype-level rubrics provide more stable evaluation standards across instances, reducing reward drift caused by independently generated instance-level rubrics.

w_p	MATH500	AIME25	BFCL (Overall)
0.1	92.1	51.7	33.3
0.3	95.8	63.3	51.3
0.5	91.0	51.7	36.1
0.7	82.7	33.4	29.9

Table 6: Ablation study on the weighting between process-level and outcome-level rewards, on Qwen3-4B-Instruct-2507. The outcome reward weight is set to $1 - w_p$.

6.3 Ablation on Reward Weighting.

We study the effect of balancing process-level and outcome-level rewards. Specifically, we vary the weight of the process reward $w_p \in \{0.1, 0.3, 0.5, 0.7\}$, with the outcome reward weight set to $1 - w_p$. All other training settings are kept fixed. Performance is evaluated on MATH500, AIME25, and BFCL v4 (Overall). Table 6 shows that the choice of reward weighting substantially affects both mathematical reasoning and tool-use performance. In particular, assigning a moderate weight to the process-level reward ($w_p = 0.3$) achieves the best overall trade-off across MATH500, AIME25, and BFCL v4, whereas excessive emphasis on either intermediate or final rewards leads to performance degradation.

7 Conclusion

We introduced SCRIBE, a reusable skill-rubric framework for reliable process reward modeling. By decomposing reasoning trajectories into subgoals and grounding reward evaluation in reusable *Skill Prototypes*, SCRIBE transforms open-ended LLM judging into structured process-level verification. Our results demonstrate strong performance across mathematical reasoning and tool-use benchmarks, notably improving the **AIME25** score of a 4B model from **43.3%** to **63.3%**.

Beyond performance, our analysis shows that reusable skill rubrics reduce dependence on high-quality reference answers and provide more consistent reward judgments across repeated evaluations and semantically similar reasoning behaviors. These findings suggest that reusable rubric grounding offers a scalable and robust alternative to instance-level rubric rewards for process reward modeling. Future work will explore more autonomous ways to expand and update skill-rubric libraries for broader open-domain reasoning tasks.

8 Acknowledgments

We wish to thank the anonymous reviewers for their helpful comments, feedback, and suggestions. Some experiments were conducted on the UMBC HPCF, supported by the National Science Foundation under Grant No. CNS-1920079. This material is also based on research that is in part supported by DARPA for the SciFy program under agreement number HR00112520301. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of DARPA or the U.S. Government.

9 Limitations

Although our approach yields consistent improvements on mathematical reasoning and tool-use benchmarks, several limitations remain. First, our experiments focus on small- to mid-sized instruction-tuned models; it is unclear how the proposed skill-conditioned supervision scales to substantially larger architectures or to models trained with different post-training pipelines. Second, the construction of skill prototypes and routing relies on clustering heuristics and judge-annotated data, which may introduce biases or limit adaptability in domains with highly diverse or ill-defined skill boundaries. Finally, our evaluation primarily targets structured reasoning and tool-use tasks; the effectiveness of the approach for open-ended generation or non-instrumental reasoning remains an open question.

10 Ethics

This work relies on publicly available datasets and uses an LLM-based judge accessed via the OpenAI API for reward evaluation (OpenAI, 2024). We do not access, attempt to access, or infer any proprietary training data or internal components of the underlying models. All experiments are conducted using standard model inference and optimization procedures, without collecting or processing personal or sensitive user data.

Licenses and Intended Use. All datasets and benchmarks are used in accordance with their released terms and intended research purposes. We do not redistribute raw datasets or proprietary

model outputs. Any derived artifacts, including probing statistics and trajectory analyses, are intended only for research and evaluation.

Artifact Documentation. Our experiments have a significant focus on English mathematical reasoning benchmarks. We additionally evaluate tool use on BFCL, while neither setting includes demographic or user-centered data.

Risks. The datasets used in our experiments are sourced from publicly available benchmarks and may contain unintended biases, errors, or harmful language. While our method aims to improve the consistency of reward signals during training, it could potentially amplify biases present in the judge or training data if deployed without additional safeguards. We also use GPT5 to assist with minor grammatical corrections in this paper.

References

- Sanjiban Choudhury. 2025. Process reward models for llm agents: Practical framework and directions. *arXiv preprint arXiv:2502.10325*.
- Murtaza Dalal, Tarun Chiruvolu, Devendra Chaplot, and Ruslan Salakhutdinov. 2024. Plan-seq-learn: Language model guided rl for solving long horizon robotics tasks. *arXiv preprint arXiv:2405.01534*.
- Guanting Dong, Yifei Chen, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Yutao Zhu, Hangyu Mao, Guorui Zhou, Zhicheng Dou, and Ji-Rong Wen. 2025. Tool-star: Empowering llm-brained multi-tool reasoner via reinforcement learning. *arXiv preprint arXiv:2505.16410*.
- Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. 2025. Rubrics as rewards: Reinforcement learning beyond verifiable domains. *arXiv preprint arXiv:2507.17746*.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- Yuxuan Jiang, Dawei Li, and Francis Ferraro. 2026. [DRP: Distilled reasoning pruning with mathematical skill-aware step decomposition for efficient large reasoning models](#). In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 4020–4039, San Diego, California, United States. Association for Computational Linguistics.
- Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhat-tacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu,

- et al. 2025. From generation to judgment: Opportunities and challenges of llm-as-a-judge. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 2757–2791.
- Renhao Li, Jianhong Tu, Yang Su, Yantao Liu, Fei Huang, Hamid Alinejad-Rokny, Derek F. Wong, Junyang Lin, and Min Yang. 2026. [ToolRM: Towards agentic tool-use reward modeling](#). In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 8613–8640, San Diego, California, United States. Association for Computational Linguistics.
- Zihan Lin, Xiaohan Wang, Jie Cao, Jiajun Chai, Guojun Yin, Wei Lin, and Ran He. 2025. Rest: Reshaping token-level policy gradients for tool-use large language models. *arXiv preprint arXiv:2509.21826*.
- Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, et al. 2024. Toolace: Winning the points of llm function calling. *arXiv preprint arXiv:2409.00920*.
- Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*.
- OpenAI. 2024. Openai api documentation. <https://platform.openai.com/docs>. Accessed via official OpenAI API.
- Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. 2025. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*.
- Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiushi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*.
- Jun Rao, Xuebo Liu, Hexuan Deng, Zepeng Lin, Zixiong Yu, Jiansheng Wei, Xiaojun Meng, and Min Zhang. 2026. [Dynamic sampling that adapts: Self-aware iterative data persistent optimization for mathematical reasoning](#). In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 28337–28350, San Diego, California, United States. Association for Computational Linguistics.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.
- Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. 2024. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*.
- Lucy Xiaoyang Shi, Joseph J Lim, and Youngwoon Lee. 2022. Skill-based model-based reinforcement learning. *arXiv preprint arXiv:2207.07560*.
- Fengwei Teng, Zhaoyang Yu, Quan Shi, Jiayi Zhang, Chenglin Wu, and Yuyu Luo. 2025. Atom of thoughts for markov llm test-time scaling. *arXiv preprint arXiv:2502.12018*.
- Tong Wu, Yang Liu, Jun Bai, Zixia Jia, Shuyi Zhang, Ziyong Lin, Yanting Wang, Song-Chun Zhu, and Zilong Zheng. 2025. Native parallel reasoner: Reasoning in parallelism via self-distilled reinforcement learning. *arXiv preprint arXiv:2512.07461*.
- Kun Xiang, Zhili Liu, Zihao Jiang, Yunshuang Nie, Kaixin Cai, Yiyang Yin, Runhui Huang, Haoxiang Fan, Hanhui Li, Weiran Huang, et al. 2025. Can atomic step decomposition enhance the self-structured reasoning of multimodal large models? *arXiv preprint arXiv:2503.06252*.
- Ningning Xu, Yuxuan Jiang, and Shubhashis Roy Dipta. 2025. Learning how to use tools, not just when: Pattern-aware tool-integrated reasoning. In *The 5th Workshop on Mathematical Reasoning and AI at NeurIPS 2025*.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.
- Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.
- Zixiong Yu, Jun Rao, Guhan Chen, Songtao Tian, Bohan Li, Jiansheng Wei, Min Zhang, and Xiaojun Meng. 2026. [Mathagent: Adversarial evolution of constraint graphs for mathematical reasoning data synthesis](#). *Preprint*, arXiv:2604.11188.
- Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, et al. 2025. The landscape of agentic reinforcement learning for llms: A survey. *arXiv preprint arXiv:2509.02547*.
- Yulai Zhao, Haolin Liu, Dian Yu, Sunyuan Kung, Meijia Chen, Haitao Mi, and Dong Yu. 2025. One token to fool llm-as-a-judge. *arXiv preprint arXiv:2507.08794*.

A Reward Calibration and Consistency Analysis

This appendix documents the prompts and diagnostics used to calibrate the reward model. Section A.1

presents the prompt variants used for mathematical and tool-use trajectory decomposition, Section A.2 describes subgoal-level reward calibration, and Section A.3 explains how fixed anchor subgoals are used to monitor reward consistency.

A.1 Prompt Variants

Prompt Variant M1 (Narrative Decomposition).

You are given a complete solution to a math problem. Analyze the solution and identify the key intermediate subgoals that are required to reach the final answer. For each subgoal, briefly describe the mathematical objective being achieved and the primary reasoning skill involved. Focus on meaningful reasoning stages rather than low-level algebraic steps. The subgoals should be ordered and together sufficient to solve the problem.

(Example output omitted.)

Prompt Variant M2 (Coverage-Constrained Segmentation).

You will receive a solution trajectory for a math problem. Segment the entire solution into a sequence of subgoals such that every part of the trajectory belongs to exactly one subgoal. Each subgoal should correspond to a coherent reasoning objective. For each subgoal, identify the dominant mathematical skill applied. Ensure no overlap or omission across subgoals.

(Example output omitted.)

Results. Across math problems, both prompt variants recover highly consistent subgoal structures. The ordering and skill attribution of subgoals largely agree with those produced by the main prompt, with high rank-level agreement in reward scores. This indicates that subgoal extraction and subsequent reward evaluation are not sensitive to prompt phrasing.

Prompt Variant T1 (Intent-Oriented Decomposition).

You are given a multi-turn, tool-using interaction. Identify the sequence of subgoals that the system completes in order to satisfy the user’s request. Each subgoal should describe a concrete intent or intermediate objective and the primary capability required to achieve it. Subgoals may span multiple turns and should reflect mid-level planning rather than surface actions.

(Example output omitted.)

Prompt Variant T2 (Failure-Aware Segmentation).

Given a tool-using conversation trajectory, segment it into subgoals such that the entire interaction is covered. If the trajectory includes partial failures, corrections, or fallback behaviors, include them as explicit subgoals. For each subgoal, identify the dominant tool-use or reasoning skill involved.

(Example output omitted.)

Results. For tool-using tasks, both prompt variants produce subgoal partitions that closely match the main prompt. Reward scores assigned based on these subgoals remain stable across prompts, with only minor variations in boundary placement for a small number of tool-related steps. Overall, the judge exhibits robust agreement across prompt formulations.

A.2 Subgoal-Level Reward Calibration

The LLM-based judge assigns discrete reward scores in $\{0, 1, 2, 3\}$ to each subgoal. To improve alignment between reward magnitude and actual correctness, we calibrate rewards using empirical subgoal-level outcome statistics.

For each subgoal type, we estimate the empirical success rate of final task completion conditioned on the assigned reward score. If a lower raw reward (e.g., 2) corresponds to a success rate comparable to a higher reward (e.g., 3), the reward is calibrated upward. Conversely, rewards that systematically overestimate correctness are adjusted downward. Calibration is implemented via a subgoal-specific lookup table, preserving relative ordering while improving reward validity.

Human Agreement. We uniformly sample 100 subgoal instances across all reward levels and manually verify whether the assigned rewards reasonably reflect subgoal quality. No systematic disagreement is observed, providing a sanity check that the reward scores align with human judgment.

A.3 Reward Consistency via Anchor Subgoals

To assess reward consistency over time, we select 20 fixed anchor subgoals for each task type. Each anchor subgoal is re-evaluated five times at different checkpoints during training.

The assigned rewards are highly stable. Only two anchor instances in tool-use tasks receive slightly different scores across evaluations, while all remaining anchor subgoals retain identical rewards. This suggests that the judge’s evaluation

criteria remain consistent over time without noticeable drift.

B Router Evaluation

We evaluate the Router on held-out judge-annotated trajectories to assess its ability to recover mid-level reasoning structure. The annotated data are split into training and test sets with a 9:1 ratio. Evaluation is performed on the test split and focuses on three aspects: step segmentation, skill prediction, and Skill Prototype retrieval.

The Router takes as input a problem and its corresponding raw reasoning trajectory, and outputs an ordered sequence of ⟨subgoal, skill, step⟩ tuples together with the associated Skill Prototype. All reported metrics are computed by comparing the Router outputs against judge-provided annotations.

Step Segmentation Accuracy. We evaluate step prediction using span-level exact match (EM). A predicted step is considered correct if its start and end positions exactly match the annotated step span in the original trajectory. This metric directly measures whether the Router correctly partitions the trajectory into non-overlapping reasoning segments.

Skill Prediction Accuracy. For each predicted subgoal, we evaluate whether the associated skill label matches the annotated skill. We report classification accuracy over all subgoals in the test set.

Skill Prototype Retrieval Accuracy. We evaluate whether each subgoal is routed to the correct Skill Prototype. A prediction is considered correct if the retrieved prototype matches the annotated prototype. This metric reflects the Router’s effectiveness in providing correct skill-level context for reward evaluation.

Metric	Accuracy (%)
Step Segmentation (EM)	94.6
Skill Prediction	95.3
Skill Prototype Retrieval	98.6

Table 7: Router evaluation results on held-out annotated trajectories. Step segmentation is measured by span-level exact match, while skill prediction and prototype retrieval are measured by classification accuracy.

C Additional Training Details

We provide additional details on GRPO training and stability diagnostics to facilitate reproducibility.

GRPO Configuration. The student model is optimized using GRPO with skill-conditioned process-level rewards. For mathematical reasoning, we sample 8 rollouts per problem over 10k training problems. We include a KL regularization term to constrain policy updates and an entropy bonus to encourage exploration. Reward shaping is applied to normalize process-level scores. Unless otherwise specified, the same optimization settings are used across all methods.

Optimization Settings. We use a fixed learning rate and batch size throughout training, with a local batch size of 128. All hyperparameters are selected from standard ranges used in prior GRPO and RLHF-style training and are kept constant across baselines.

Training Stability. To monitor training stability, we track reward statistics over time, including mean reward and reward variance. We observe no evidence of reward collapse during training.

Cost The overall cost of all experiments was approximately USD 1,800.

D Prompts for Subgoal and Step Decomposition

We use structured prompts to decompose each problem or trajectory into mid-level subgoals and associated skills, which serve as the basis for our skill-aware supervision and analysis.

Subgoal & Skill Extraction Prompt (Math)

I will give you a math problem. Break the solution into several subgoals, where completing these subgoals in order is sufficient to solve the whole problem. For each subgoal, explicitly name the main skill required, and give one short step showing how it is applied.

Use this format: (1) subgoal: <what this subgoal accomplishes> skill: <main skill needed> step: <one short sentence>

*Example: **subgoal:** maximize $u = (x + 4)y$ subject to $x^2 + y^2 = 1$*

***skill:** optimization with constraint (Lagrange multipliers or trigonometric substitution)*

***step:** set $x = \cos \theta$, $y = \sin \theta$, so*

$u(\theta) = (\cos \theta + 4) \sin \theta = \frac{1}{2} \sin 2\theta + 4 \sin \theta$,

and maximize over θ .

Now apply this style to the following problem:

Example Output (Tool-Using Trajectory)

You will receive a multi-turn, tool-using conversation trajectory. The user's final goal is achieved step by step through the completion of multiple subgoals. Subgoals may span multiple turns and should reflect MID-LEVEL reasoning or planning.

Your task is to identify all subgoals that are carried out in order to accomplish the user's final goal.

For each subgoal: Specify the primary skill required to complete this subgoal. Identify the corresponding step(s) in the original trajectory. Each subgoal must correspond to a single contiguous, unmodified subpart of the original trajectory. The entire trajectory must be partitioned into such subparts with no overlap and no omission: every part of the trajectory should belong to exactly one subgoal.

If the task is partially completed or fails, include subgoals that capture request decomposition, recognition of capability limitations, or fallback handling.

*Example output: (1) subgoal: get the top market trend in us skill: situational assessment from structured signals step: Here are the top Market Trends in the US right now: 1. **S&P 500**: The Standard & Poor's 500 Index is a market-capitalization-weighted index of the 500 largest U.S. publicly traded companies. Its current value is 4172.80 with a percentage change of +0.68%.*

*2. **DOW J**: The Dow Jones Industrial Average is a price-weighted average of 30 blue-chip stocks that are generally the leaders in their industry. Its current value is 34479.60 with a percentage change of +0.47%.*

*3. **NASDAQ**: The NASDAQ Composite is a broad-based capitalization-weighted index of stocks in all three NASDAQ tiers: Global Select, Global Market and Capital Market. Its current value is 13691.30 with a percentage change of +0.90%. This information can help you make informed decisions about your investment plans.*

E Cluster and Prototype Examples

Math Prototype In this section, we show two examples of our skill prototype (Non-cherry picking).

(III) Skill Prototype (Mathematical Reasoning)

Skill Name: Bound-Based Conclusion and Synthesis

Knowledge Scope: Upper/lower bounding; extremal principles; attainability (tightness) arguments; case coverage; domain constraints; logical implication.

Applicable Scenario: Subgoals that *terminate* a reasoning chain by drawing a final conclusion from previously derived constraints (e.g., impossibility via bounds, identifying a maximal value, selecting a correct option after logical elimination), including cases where equality/tightness or feasibility must be justified.

Canonical Reasoning Pattern:

1. Identify relevant intermediate results (bounds, constraints) that the conclusion must follow from.
2. Determine if the subgoal requires an implication or a tightness claim (e.g., "max equals bound").
3. If concluding an extremum, check **tightness**: reference a construction or equality condition.
4. Verify domain restrictions and case completeness (e.g., integer ranges, geometric feasibility).
5. State the final conclusion succinctly (value, choice, or maximal attainable level).

Judging Rubric (0–3 Step Score):

- **3 (Correct & Complete):** Conclusion follows rigorously from prior results; addresses tightness/feasibility.
- **2 (Minor Flaw):** Correct strategy and conclusion, but with minor slips (arithmetic/wording) or obvious-but-unstated justifications.
- **1 (Major Logical Gap):** Correct skill type, but substantial omissions (e.g., ignores a case) making the conclusion unsupported.
- **0 (Wrong Skill):** Incompatible/premature conclusion or complete misapplication of bounding logic.

Common Traps & Scoring Mapping :

Common Trap	Score	Reasoning for Penalty
Boundary Leak (e.g., leaving $c < 44.75$ without floor)	2	Method is correct, but the final conclusion lacks the discrete precision required by the domain.
Implicit Tightness (Assuming max is	1	A major logical gap: in competition math, a bound is not a

Representative Reference Step: From "for $n \geq 5$, $6n \leq 2^n$ (impossible)" and "levels $n \leq 4$ are feasible", conclude the max level is 4.(more omitted)

(IV) Skill Prototype (Tool-Using Reasoning)

Skill Name: Capability Limitation Handling and Tool-Mediated Fallback Guidance

Knowledge Scope: Tool selection and parameterization; structured API invocation; tool output parsing; user-facing summarization; uncertainty calibration; data access limitations.

Applicable Scenario: Tool-using tasks where the agent must (i) invoke a tool for analysis or retrieval, (ii) interpret and present tool outputs, and/or (iii) recognize missing access, unavailable data, or capability limits and provide actionable fallback guidance (e.g., how to obtain required inputs, what can be done instead, or how the agent can proceed once inputs are provided).

Canonical Reasoning Pattern:

1. **Goal & Constraints Identification:** Restate the user intent and identify required inputs (entity IDs, time range, text to analyze) and constraints (access, tool availability, policy limits).
2. **Tool Selection & Invocation:** Choose the correct tool and issue a well-formed call with task-relevant parameters (e.g., shareuid, from/to; text=).
3. **Structured Output Interpretation:** Parse returned fields and map them to the user request (e.g., roa_ratio → ROA for FY2025; sentiment label + keywords).
4. **Limitation Detection & Fallback:** If the request cannot be completed (missing access/data/tool), explicitly state the limitation, offer concrete next steps (how to obtain the data / what to provide), and propose safe alternatives.

Judging Rubric (0–3 Step Score):

- **3 (Correct & Complete):** Correctly identifies constraints...(ommitted)
- **2 (Minor Flaw):** Overall correct skill and intent, but with minor issues (slightly suboptimal parameters, small formatting problems, or partially specified fallback that is still usable).
- **1 (Major Logical Gap):** Correct general direction (tool use or limitation handling) but substantial problems: wrong/missing critical parameters, misreading key fields, vague or non-actionable fallback, or mixing speculation with tool outputs.
- **0 (Wrong Skill / Unsafe / Hallucination):** Uses the wrong tool or fabricates tool outputs/data, claims completion despite missing access, or provides unsafe/irrelevant guidance.

Common Traps & Scoring Mapping:

Common Trap	Score	Reasoning for Penalty
Hallucinated Tool Output (inventing data without a tool result)	0	Breaks tool grounding: the output is unverifiable and violates the tool-mediated protocol.
Non-Actionable Limitation (only says “I can’t” without next steps)	1	Recognizes a limitation but does not complete the subgoal of providing feasible fallback guidance.
Over-Refusal Despite Feasible Alternative (refuses when public steps or user-provided inputs could proceed)	2	Limitation acknowledged, but the response underutilizes available actions; still partially helpful.

Representative Reference Step: If a user requests ROA for shareuid=6789 for FY2025, invoke Financial Fundamentals API(shareuid=6789, from="2025-01-01", to="2025-12-31"), then report ...(ommitted) (more ommitted)