Extended Abstract

Motivation. Reinforcement learning from verifier rewards is attractive for mathematical reasoning because final answers can often be checked automatically. In the Countdown arithmetic task, however, the standard verifier only evaluates the final expression: it gives strong signal when the model writes a valid expression that reaches the target, but it gives little credit-assignment information about which intermediate search steps were legal, useful, repeated, or misleading. This is especially limiting for small language models trained with online RLOO, where many sampled groups are either all correct or all wrong and therefore provide weak within-prompt contrast.

Method and novelty. We study whether Countdown's symbolic structure can support process-aware RLOO without a learned process reward model. We replace free-form reasoning with structured search traces containing explicit number-pool states, attempted operations, and occasional backtracking. A deterministic depth-first-search generator creates supervised traces, and a programmatic verifier parses generated traces to check state transitions, invalid operations, repeated actions, and whether the trace reaches the target. Our extension is a controlled study of programmatic process rewards: instead of asking whether a neural verifier can judge reasoning steps, we ask whether exact symbolic process feedback is sufficient to improve RLOO on a small language model.

Implementation. We initialize structured-trace SFT from the base Qwen2.5-0.5B model rather than from the completed milestone SFT/RLOO checkpoints, because the extension changes the output schema. We generate 2,355 training traces and 453 test traces in the best proof-plus-light-backtracking dataset, train for three SFT epochs with EOS appended during tokenization, and then run 50-step structured RLOO signal experiments with matched rollout budgets. We evaluate all headline results with the same standard no-stop Countdown evaluator used for the baselines, using 50 held-out prompts and 16 sampled completions per prompt.

Results. The structured-trace SFT warm start reaches pass@1 0.273750 and pass@16 0.700000, making it competitive with the standard SFT baseline while exposing the model's search process. Outcome-only structured RLOO is the strongest clean extension result: it improves pass@1 from 0.273750 to 0.523750 while preserving pass@16 at 0.700000. In contrast, the best clean process-aware reward, a success-gated trace bonus, reaches pass@1 0.466250 and pass@16 0.540000. Other process-reward ablations, including dense trace-validity rewards, entropy-regularized trace bonuses, group-aware shaping, difficulty-aware sampling, and effort/hardness bonuses, also underperform the outcome-only structured RLOO baseline.

Discussion and conclusion. The negative process-reward result is the main scientific finding. Structured traces reveal that the model often reaches useful intermediate states, but fails to convert those states into a legal final expression over the original input numbers. Dense process rewards can also change RLOO's within-group rankings in harmful ways: they may reinforce clean but wrong traces, collapse diversity toward short add/subtract-heavy solutions, or increase trace complexity without fixing final-answer grounding. We conclude that structured traces are valuable for inspectability and compatible with RLOO, but future process rewards should preserve expression provenance or add a trace-to-answer repair mechanism before rewarding local process effort more aggressively.

Structured Search Traces for Process-Aware Countdown Training

Yucheng Huang
Stanford University
yuchengh@stanford.edu

Parth Sheth
Stanford University
parth2@stanford.edu

Abstract

Final-answer verifiers provide scalable reward signals for mathematical language-model training, but they do not identify which intermediate reasoning steps were valid or useful. We investigate this issue on the Countdown arithmetic task by training a small language model to emit structured search traces and by augmenting RLOO with deterministic process rewards over those traces. Our method uses a depth-first-search generator to produce state/action/backtracking traces for SFT, then compares outcome-only structured RLOO against several process-aware reward variants that check trace validity, target-reaching states, group structure, effort, and prompt hardness. Structured traces are a viable warm start: proof-plus-light-backtracking SFT reaches pass@1 0.273750 and pass@16 0.700000, and outcome-only structured RLOO improves pass@1 to 0.523750 while preserving pass@16. However, process-aware rewards do not improve over this baseline; the strongest reaches pass@1 0.466250 and pass@16 0.540000. Diagnostics show that process rewards reduce useful coverage by collapsing diversity or by increasing final-expression grounding failures, where the trace reaches useful intermediate states but the final answer uses generated intermediate values rather than the original numbers. These results suggest that process-verifiable traces are useful for analysis, but effective process RL for symbolic reasoning must preserve expression provenance and final-answer grounding.

1 Introduction

Verifier-based reinforcement learning is a natural fit for mathematical reasoning tasks: if a final answer can be checked automatically, a language model can be fine-tuned without human preference labels for every generated solution. This is the appeal of applying RL fine-tuning methods such as RLOO to Countdown, where a response is correct if its final arithmetic expression uses each input number exactly once and evaluates to the target. The weakness is that the reward is sparse. A rollout with several correct intermediate transformations but a final reconstruction mistake receives nearly the same score as an invalid rollout, and an all-wrong RLOO group gives little information about which sampled trajectory was closer to success.

Our project asks whether Countdown's symbolic structure can provide process supervision without a learned process reward model. Prior work has shown the value of process supervision for mathematical reasoning Lightman et al. (2024) and has explored automated process verifiers Setlur et al. (2025), while search-trace training has shown that language models can learn from explicit traces of exploration and backtracking Gandhi et al. (2024); Moon et al. (2025). Countdown gives us a simpler but sharper test bed: every intermediate state is a multiset of remaining numbers, and every action is a deterministic arithmetic transition. This lets us construct a fully programmatic process verifier.

We study three questions. First, can a 0.5B parameter language model learn a parseable structured search format for Countdown? Second, does such a structured SFT warm start remain compatible

with online RLOO under the original sparse final verifier? Third, do programmatic process rewards improve coverage, or do they introduce new failure modes?

Our main conclusion is mixed but informative. Structured traces are successful as an interface: the best structured SFT warm start is competitive with the standard SFT baseline, and outcome-only structured RLOO substantially improves pass@1. But our process-aware reward variants underperform outcome-only structured RLOO. The failure analysis shows that local trace validity is not the main remaining bottleneck. The model often produces useful trace states but fails to reconstruct a final expression using only the original input numbers. The central lesson is therefore not that process feedback is useless, but that process rewards must be designed around final-answer grounding and rollout diversity, not only local step validity.

2 Related Work

RL fine-tuning and RLOO. RLHF methods optimize language models using reward signals derived from human preferences or learned reward models Ouyang et al. (2022). RLOO and related REINFORCE-style approaches revisit simpler policy-gradient estimators for language-model alignment, using multiple samples per prompt and a leave-one-out baseline to reduce variance Ahmadian et al. (2024). Our implementation follows this group-relative structure, but our reward is programmatic and task-specific.

Process supervision. Process supervision provides feedback on intermediate reasoning steps rather than only final outcomes. Lightman et al. show that step-by-step verification can improve mathematical reasoning Lightman et al. (2024), but such supervision often requires expensive annotations. Setlur et al. propose automated process verifiers that reward progress toward correct solutions Setlur et al. (2025). Our work studies a special symbolic setting where process verification is exact and hand-coded: each Countdown action can be checked by deterministic arithmetic and multiset updates.

Learning to search in language. Stream-of-search training flattens search trajectories, including exploration and backtracking, into language-model training data Gandhi et al. (2024). Guided reinforced self-training further incorporates search guidance during RL Moon et al. (2025). Our structured traces are inspired by this view, but our goal is not only to imitate search traces. We use traces as an interface for online RLOO rewards and diagnostics on a pretrained small model.

Exploration and test-time reasoning. Sparse final rewards can make exploration difficult, particularly when many sampled groups contain no successful rollouts. Recent work studies exploration-aware preference optimization and test-time compute allocation for language models Xie et al. (2024); Qu et al. (2025). Our experiments provide a concrete example of this issue: process rewards can make high-probability correct traces cleaner while reducing the diversity needed for pass@16 coverage.

3 Method

3.1 Countdown Scoring

Each Countdown prompt gives a target integer and a small multiset of allowed numbers. The model must produce a final expression inside `<answer>` tags using each input number exactly once. We use the standard project verifier: a response receives reward 1.0 if the final extracted expression is valid and equals the target, 0.1 if it has answer tags but the expression is invalid or wrong, and 0.0 if no answer is found. The verifier extracts the last `<answer>...</answer>` span in the raw generated text, so natural termination matters under the standard no-stop evaluation policy.

3.2 Structured Trace Format

The extension changes the model output from free-form reasoning to a line-based search trace. A trace contains the current number pool, attempted operations, optional backtracking, and a final answer:

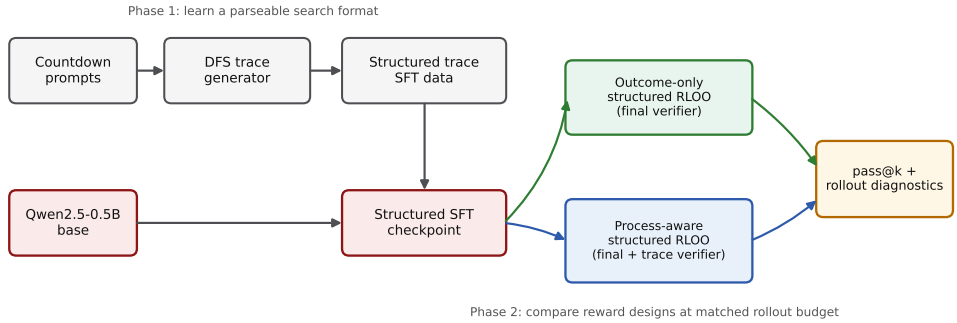

Figure 1: Method overview. We first generate structured traces with a DFS solver and use them for SFT from the base model. We then compare outcome-only structured RLOO against process-aware structured RLOO under matched rollout budgets, evaluating both pass@k and rollout diagnostics.

```
<think>
state [2, 3, 7, 8]
try 8 - 2 = 6
state [3, 6, 7]
try 7 - 3 = 4
state [4, 6]
try 4 * 6 = 24
state [24]
</think>
<answer> (8 - 2) * (7 - 3) </answer>
```

Listing 1: Structured trace format for Countdown.

This format makes the reasoning process parseable. A process verifier can check whether both operands are present in the current pool, whether the arithmetic result is correct, whether the next state matches the multiset update, whether an action repeats, and whether the final trace state reaches the target.

3.3 Structured Trace SFT

Because the structured output schema differs from the completed milestone SFT and RLOO checkpoints, we initialize the extension SFT stage from the base Qwen2.5-0.5B model rather than from a standard-answer checkpoint. The trace data is generated by a DFS solver over Countdown number pools. Every generated row is validated by both the strict final-answer verifier and a step-by-step transition checker. We experimented with proof-only traces, backtrack-heavy traces, direct-answer mixtures, and proof-plus-light-backtracking traces. The best warm start uses proof traces for most rows and one-backtrack traces for a small subset, giving 2,355 train rows and 453 test rows.

One important implementation lesson is that teacher-forced token accuracy was not enough. Several early structured SFT models learned local trace tokens but failed during autoregressive rollout, especially by continuing after a correct `</answer>` tag. We therefore append the tokenizer EOS during SFT tokenization while keeping the dataset text clean. This reduced post-answer continuation in the bounded-backtrack setting from 787 continuing outputs out of 787 answer-closed outputs to 0 continuing outputs out of 798 answer-closed outputs.

3.4 RLOO Training

For each training prompt, RLOO samples a group of G completions from the current policy. Given rewards $r_1, \ldots, r_G$, the leave-one-out advantage for sample i is

$$A_i = r_i - \frac{1}{G-1} \sum_{j \neq i} r_j. \tag{1}$$

Table 1: Process-aware reward and sampler variants. Internal IDs are included only for traceability to the code and experiment logs.

Paper label	Internal ID	Hypothesis tested
Outcome-Only Structured RLOO	final-only	Structured traces can improve under the sparse final verifier alone.
Dense Trace-Validity Reward	process-v1	Dense rewards for valid transitions and matching states improve credit assignment.
Success-Gated Trace Bonus	process-v2	Process credit should apply only when the strict final answer is already correct.
Entropy-Regularized Trace Bonus	process-v2 entropy	Stronger entropy can recover diversity lost by process shaping.
Penalty-Guarded Trace Reward	process-v3	Small capped penalties can reduce malformed or ungrounded traces.
Outcome-Only Exploration	final-only explore	Larger groups, higher temperature, and stronger entropy can expand coverage without changing rewards.
Group-Aware Trace Bonus	process-v4	All-wrong groups should remain outcome-only to avoid ranking wrong traces.
Difficulty-Aware Outcome-Only RLOO	difficulty sampler	Sampling mixed-success prompts can expand coverage without changing rewards.
Effort-Gated Trace Bonus	process-v5	Correct traces with extra unique valid effort should be preferred over short traces.
Effort + Hardness Bonus	process-v6	Correct hard-prompt samples need a larger margin over wrong samples.

The policy-gradient update uses the sequence log probability of the response, optional clipped importance weights comparing the update model to the vLLM sampling log probability, entropy regularization, and an optional KL penalty against the reference model. The extension reuses this RLOO update and changes only the warm start, output format, sampling diagnostics, and reward computation.

During structured RLOO development we found and fixed a token-alignment bug in the importance weights: vLLM cumulative log probabilities were computed over sampled token IDs, while the update worker previously recomputed log probabilities after retokenizing stop-truncated text. The clean experiments reported here pass vLLM prompt and response token IDs directly into the update worker. Pre-fix structured RLOO runs are not used for scientific claims.

3.5 Process Reward Variants

The process verifier returns both a scalar reward and diagnostics. We evaluated several reward designs, using descriptive labels in the paper:

4 Experimental Setup

We also report the completed standard-format 200-step RLOO milestone result for context. The main structured extension comparison uses 50-step signal runs to keep outcome-only and process-aware structured RLOO at a matched rollout budget.

5 Results

5.1 Structured Trace SFT

Figure 2 shows that the structured format itself is a nontrivial part of the extension. Proof-only traces underperform badly because rollouts often repeat state lines without making valid attempts. Backtracking helps the model learn an action vocabulary and increases coverage at larger k, but too much backtracking causes long unstable traces. A more detailed dataset timeline is provided in Appendix A. The main lesson is that the warm start needs a consistent action grammar, EOS supervision, controlled solution diversity, and a low but nonzero amount of backtracking. The proof-

Table 2: Experimental setup for the structured-trace extension.

Category	Setting
Base model	Qwen2.5-0.5B base model.
Task and evaluation set	Countdown arithmetic; 50 held-out prompts from `asingh15/countdown_tasks_3to4`.
Evaluation samples	16 completions per prompt, 800 total samples per run.
Headline metric	Corrected pass@k estimator over prompts; score 1.0 is counted as correct.
Evaluation policy	Standard no-stop Countdown evaluator, matching completed SFT/IPO/R-LOO baselines.
Structured SFT	3 epochs, learning rate 5×10^{-5}, batch size 64, gradient accumulation 8, EOS appended during tokenization.
RLOO signal runs	50 updates, batch size 64, group size 8, learning rate 10^{-5}, max response length 512.
Regularization	Entropy coefficient 0.001 and KL coefficient 0.001 unless explicitly ablated.
Diagnostics	Any-correct prompts, all-wrong prompts, unique responses, answer failure categories, trace parser counts, and multiplication/division usage.

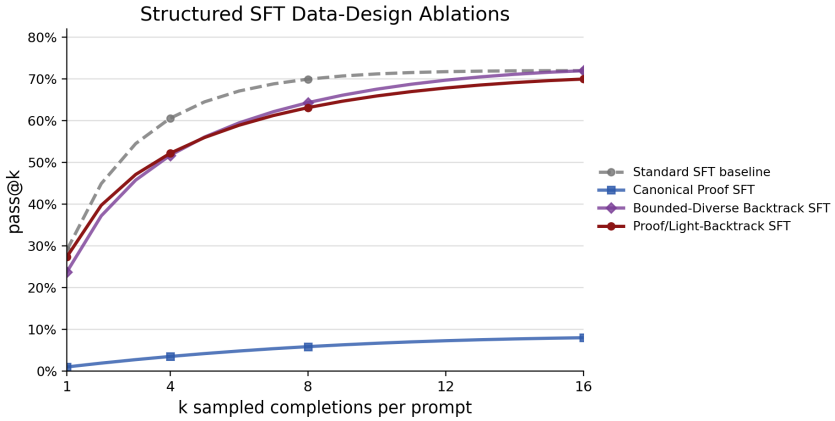

Figure 2: Structured-trace SFT data-design ablations. Proof-only traces are too unstable for rollout. Backtrack-heavy traces improve pass@16 but encourage over-searching. Proof plus light backtracking gives the best warm start, reaching pass@1 0.273750 and pass@16 0.700000.

plus-light-backtracking dataset is the best compromise: it recovers the structured action grammar and gives a viable RLOO warm start.

5.2 Main Quantitative Comparison

Table 3 and Figure 3 show the central result. Outcome-only structured RLOO improves pass@1 from 0.273750 to 0.523750 while preserving pass@16 at 0.700000. This means the structured SFT checkpoint remains compatible with sparse final-answer RLOO. However, it does not expand the solved-prompt frontier: both structured SFT and outcome-only structured RLOO solve at least one sample for 35 of 50 evaluation prompts. The improvement mostly makes already reachable solutions more likely.

The strongest clean process-aware reward is the Success-Gated Trace Bonus, which gives a small bonus only to strict-correct samples whose parsed trace reaches the target. It fixes the most obvious reward-hacking failure of dense trace-validity rewards, but it still drops to pass@16 0.540000. This suggests that process shaping can reduce coverage even when wrong final answers receive no positive process bonus.

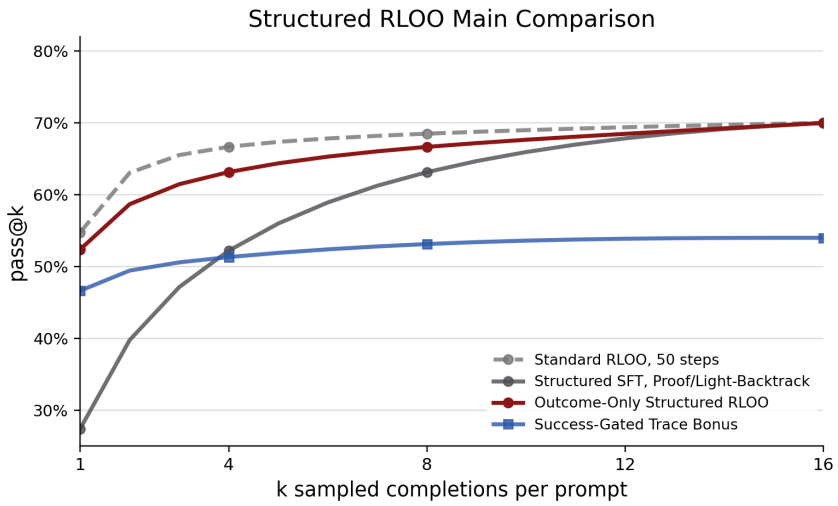

Figure 3: Matched-budget structured-trace extension pass@k. Outcome-only structured RLOO improves the structured SFT warm start and approaches the 50-step standard RLOO reference. The best process-aware variant underperforms outcome-only structured RLOO, especially at larger k, where coverage matters.

Table 3: Main pass@k results under the standard no-stop evaluator. The standard RLOO 50-step reference is the matched-budget comparison for structured 50-step signal runs; the 200-step RLOO row is included as the completed milestone baseline.

Method	pass@1	pass@4	pass@8	pass@16
SFT baseline	0.290000	0.606011	0.699764	0.720000
IPO baseline	0.392500	0.678187	0.742471	0.760000
Standard RLOO, 50 steps	0.547500	0.666802	0.685049	0.700000
Standard RLOO, 200 steps	0.640000	0.741956	0.759217	0.780000
Structured SFT, proof+light backtracking	0.273750	0.522110	0.631473	0.700000
Outcome-Only Structured RLOO	0.523750	0.631615	0.666648	0.700000
Success-Gated Trace Bonus	0.466250	0.513176	0.531333	0.540000

5.3 Reward and Sampler Ablations

The ablations in Table 4 and Figure 4 test a sequence of hypotheses. Dense Trace-Validity Reward rewards locally valid transitions and matching states, but this gives high relative reward to clean traces whose final answer is wrong. Success-Gated Trace Bonus removes positive process credit for wrong answers, but the policy still collapses toward short, safe, mostly addition/subtraction traces. Outcome-Only Exploration isolates whether larger groups, higher temperature, and stronger entropy are enough; they are not, and this run loses useful operation diversity. Entropy and group-aware variants recover some surface diversity but do not recover useful coverage. Effort and hardness bonuses were motivated by the hypothesis that process rewards over-reinforced short traces; they recover some operation variety, but still fail because longer traces make final-expression grounding harder.

6 Qualitative Analysis

Table 5 shows two recurring failure modes. First, some process rewards collapse useful diversity. Dense Trace-Validity Reward reduces the mean number of unique responses per prompt to 1.70 and eliminates multiplication/division usage in both traces and answers. Second, variants that preserve or recover diversity still struggle with final-answer grounding. Group-Aware Trace Bonus has nearly the same unique-response mean as outcome-only structured RLOO, but it increases invalid-number answers and solves fewer prompts at least once.

Table 4: Process-reward and sampler ablations. All rows use the structured SFT v5 warm start and 50-step signal budget.

Variant	pass@1	pass@4	pass@8	pass@16
Outcome-Only Structured RLOO	0.523750	0.631615	0.666648	0.700000
Dense Trace-Validity Reward	0.372500	0.393978	0.405333	0.420000
Success-Gated Trace Bonus	0.466250	0.513176	0.531333	0.540000
Entropy-Regularized Trace Bonus	0.436250	0.520099	0.549790	0.560000
Penalty-Guarded Trace Reward	0.361250	0.408407	0.444462	0.480000
Outcome-Only Exploration	0.396250	0.428297	0.437744	0.440000
Group-Aware Trace Bonus	0.373750	0.438242	0.475916	0.520000
Difficulty-Aware Outcome-Only RLOO	0.425000	0.517956	0.553901	0.580000
Effort-Gated Trace Bonus	0.405000	0.448011	0.465007	0.480000
Effort + Hardness Bonus	0.388750	0.450582	0.465262	0.480000

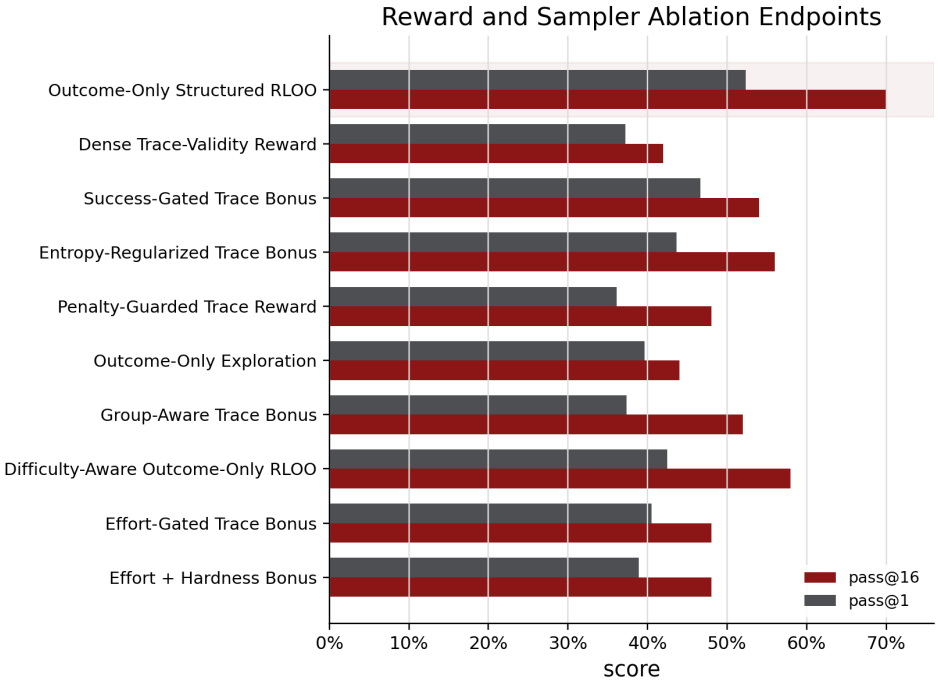

Figure 4: Endpoint view of the reward and sampler ablations. Plotting pass@1 and pass@16 together highlights the main pattern: the process-aware variants and exploration/sampling changes do not recover the pass@16 coverage of Outcome-Only Structured RLOO.

```
Target: 91
Numbers: [79, 8, 27, 47]

<think>
state [8, 27, 47, 79]
try 79 + 47 = 126
state [8, 27, 126]
try 126 - 27 = 99
state [8, 99]
try 99 - 8 = 91
state [91]
</think>
<answer> ((126 - 27) - 8) </answer>
```

Table 5: Rollout diagnostics over 800 standard-evaluation samples. These diagnostics explain why process rewards underperform outcome-only structured RLOO.

Method	Correct	Any-correct	Unique mean	Invalid nums	Trace */
Structured SFT v5	219	35/50	13.22	413	313
Outcome-Only Structured RLOO	419	35/50	8.66	274	365
Dense Trace-Validity Reward	298	21/50	1.70	107	0
Success-Gated Trace Bonus	373	27/50	3.22	116	3
Group-Aware Trace Bonus	299	26/50	8.60	354	178
Difficulty-Aware Outcome-Only RLOO	340	29/50	7.18	317	129
Effort-Gated Trace Bonus	324	24/50	5.32	357	213
Effort + Hardness Bonus	311	24/50	6.04	233	15

Listing 2: Representative trace-to-answer grounding failure. The trace reaches the target state, but the final answer uses an intermediate value instead of only original numbers.

Listing 2 illustrates the main qualitative bottleneck. The trace is locally meaningful and reaches `state [91]`, but the final expression contains `126`, a generated intermediate value. The strict Countdown verifier requires the final expression to use the original numbers exactly, so this receives only format credit. Structured traces make this failure visible, but our current process rewards do not reliably fix it.

We also observe operation collapse. For example, with target 40 and numbers $[96, 74, 94, 54]$, outcome-only structured RLOO often solves the prompt using multiplication:

$$(74 - 54) \times (96 - 94) = 40.$$

Several process-aware variants instead concentrate on addition/subtraction templates such as $(74 - 54) + (96 - 94)$, which are locally simple but miss the target. This explains why pass@16 drops: larger k rewards coverage over distinct solution families, and process rewards often shrink that coverage.

7 Discussion

The results show that process rewards must be evaluated through RLOO's within-group dynamics. Dense Trace-Validity Reward looked reasonable as a scalar formula: valid transitions and matching states should be better than malformed traces. But in all-wrong groups, rewarding the cleanest wrong trace creates a relative advantage among incorrect samples. The policy can then reinforce traces that look valid but do not solve the task.

Success-gated rewards avoid that specific problem by giving no positive process bonus to wrong final answers. However, they introduce a different selection effect. A process bonus is visible only on prompts where the current policy already samples at least one correct response. Hard prompts with all-wrong groups still provide almost no useful process signal. Over training, the policy is pushed toward already-successful short traces, especially addition/subtraction-heavy traces that are easy to express in a final answer.

The Effort-Gated Trace Bonus tests whether the issue is merely short-trace collapse. It recovers some diversity and operation variety relative to the Success-Gated Trace Bonus, but invalid-number final answers increase. This supports a more precise hypothesis: the dominant remaining problem is trace-to-answer reconstruction. Longer or more exploratory traces expose more intermediate values, and the current trace representation does not preserve the expression provenance needed to expand those values back into legal final answers.

Limitations. Most structured RLOO experiments are 50-step signal runs rather than long multi-seed sweeps, so small differences should not be overinterpreted. The experiments use one small base model and one symbolic task. The process verifier is hand-designed for Countdown and does not directly transfer to open-ended reasoning. Finally, standard no-stop evaluation is strict but comparable to the completed project baselines; stopped-at-answer diagnostics can look different and should be treated separately.

Broader impacts. Programmatic process verification can reduce the need for human step labels in symbolic domains and can make model reasoning more auditable. At the same time, our results warn against trusting process-looking traces by themselves. Optimizing local trace validity can produce cleaner but still incorrect reasoning, so process rewards should always be validated against final task correctness and coverage.

8 Conclusion

We implemented and evaluated structured search traces for process-aware RLOO on Countdown. The structured format is useful: it can be learned by a small base model, supports programmatic diagnostics, and remains compatible with sparse outcome-only RLOO. The strongest clean extension result is Outcome-Only Structured RLOO, which improves pass@1 to 0.523750 while preserving pass@16 0.700000. However, our process-aware reward variants do not beat this baseline. The main lesson is that local process validity is not sufficient; rewards must preserve diversity and final-answer grounding. Future work should add expression provenance to trace states, train a trace-to-answer repair stage, or design process rewards that explicitly check consistency between terminal trace states and legal final expressions over the original numbers.

9 Team Contributions

- **Yucheng Huang**: SFT & RLOO implementation / training, search trace format design, synthetic SFT dataset generation, programmatic process verifier, RLOO ablations training / evaluation.
- **Parth Sheth**: IPO implementation / training, RLOO reward integration, DFS Solver implementation, synthetic SFT dataset generation, programmatic process verifier.

10 AI Tools Disclosure

AI tools, including ChatGPT/Codex, were used for slight code and LaTeX assistance. We developed and verified the experiments, implementation, results, and scientific conclusions ourselves.

References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 2024. Back to Basics: Revisiting REINFORCE Style Optimization for Learning from Human Feedback in LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*.

Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D. Goodman. 2024. Stream of Search (SoS): Learning to Search in Language. arXiv:2404.03683 [cs.CL]

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's Verify Step by Step. In *International Conference on Learning Representations*.

Seungyong Moon, Bumsoo Park, and Hyun Oh Song. 2025. Learning to Better Search with Language Models via Guided Reinforced Self-Training. In *Advances in Neural Information Processing Systems*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training Language Models to Follow Instructions with Human Feedback. arXiv:2203.02155 [cs.CL]

Yuxiao Qu, Matthew Y. R. Yang, Amrith Setlur, Lewis Tunstall, Edward Emanuel Beeching, Ruslan Salakhutdinov, and Aviral Kumar. 2025. Optimizing Test-Time Compute via Meta Reinforcement Fine-Tuning. arXiv:2503.07572 [cs.LG]

Table 6: Structured-trace SFT dataset iteration. All pass@k values use the standard no-stop evaluator.

Version	Main idea	pass@1	pass@16	Main lesson
Smoke conversion	Early converted traces with formatting issues	0.000000	0.000000	Uppercase tags and answer contents such as "= target" were incompatible with the strict scorer.
Expr-only v1	Lowercase tags and expression-only answers	0.006250	0.080000	Formatting fixes helped, but converted natural-language traces were still too noisy.
Proof v1	Programmatic DFS proof-only traces	0.050000	0.300000	Clean proof traces still collapsed into repeated state lines during rollout.
Backtrack v1	Bounded backtrack traces	0.042500	0.400000	Backtrack syntax helped coverage, but no-EOS training caused post-answer continuation.
Backtrack v1 + EOS	Same data with EOS appended during SFT tokenization	0.260000	0.660000	EOS supervision was crucial for natural termination under no-stop evaluation.
Augmented proof v1	More DFS proof variants	0.000000	0.000000	More variants hurt; the model emitted repeated states and almost no valid tries.
Augmented backtrack v1	More DFS variants with backtracking	0.075000	0.400000	Backtrack grammar survived, but too much branching weakened semantic grounding.
Canonical proof v2	One canonical proof per prompt	0.010000	0.080000	Removing diversity made the policy too narrow and did not fix rollout stability.
Canonical backtrack v2	Canonical solution plus controlled distractor	0.082500	0.500000	Cleaner than augmented backtrack, but one completion per prompt removed useful variation.
Bounded-diverse backtrack v3	Multiple completions with safer distractors	0.237500	0.720000	Diversity improved pass@16, but plausible distractors caused over-searching.
Mixed v4	Direct-answer, proof, and low-frequency backtrack rows	0.043750	0.440000	Mixing direct answers under the structured prompt broke the action grammar.
Proof/light-backtrack v5	Mostly proof rows with low-frequency backtracking	0.273750	0.700000	Best warm start: stable action grammar, strong pass@1, and enough pass@16 coverage for RLOO.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. 2025. Rewarding Progress: Scaling Automated Process Verifiers for LLM Reasoning. In *International Conference on Learning Representations*.

Tengyang Xie, Dylan J. Foster, Akshay Krishnamurthy, Corby Rosset, Ahmed Awadallah, and Alexander Rakhlin. 2024. Exploratory Preference Optimization: Harnessing Implicit Q*-Approximation for Sample-Efficient RLHF. arXiv:2405.21046 [cs.LG]

A Structured SFT Dataset Iteration

Table 6 summarizes the structured SFT sequence that led to the final warm start. These runs were not just hyperparameter noise: each tested a concrete hypothesis about how to make a small model emit parseable search traces under the same strict no-stop evaluator used for the baselines.

Backtrack-v3 is especially informative because it achieved the best structured SFT pass@16, but qualitative diagnostics showed long repeated search traces with excessive backtracking. Mixed-v4 then tested whether direct-answer rows could preserve final-expression skill, but the small model averaged incompatible completion modes and often stopped emitting valid `try` lines. V5 therefore removes direct-answer rows while keeping low-frequency backtracking, which makes it the best RLOO warm start even though its pass@16 is slightly below v3.

B Additional Process-Reward Diagnostics

The process-aware reward variants were designed as hypothesis tests rather than as a single monotonic hyperparameter sweep. Dense Trace-Validity Reward tested whether local transition validity was enough to improve sparse RLOO. Success-Gated Trace Bonus then removed positive process credit for wrong final answers. Entropy-Regularized Trace Bonus, Group-Aware Trace Bonus, Difficulty-Aware Outcome-Only RLOO, Effort-Gated Trace Bonus, and Effort + Hardness Bonus each targeted one observed failure mode: diversity collapse, all-wrong group reward pollution, prompt selection, short-trace collapse, and hard-prompt margin respectively.

C Importance-Weight Alignment

The original structured RLOO signal runs before the final fix compared vLLM cumulative log probabilities over sampled token IDs against PyTorch log probabilities over retokenized stopped text. This could change terminal tokens around `</answer>` and collapse importance weights. The corrected trainer passes vLLM prompt and response token IDs directly to the update worker. A one-step smoke run after the fix had importance-weight mean 0.9711 and log-probability difference mean -0.1230, which is consistent with small numerical differences rather than sequence mismatch.