Extended Abstract

Structured Q&A Reasoning for Language Models

Motivation RL fine-tuning of reasoning models is dominated by *outcome* rewards—a single scalar that fires only when the final answer is correct. This signal is sparse and says nothing about *how* the model reasoned, so the policy can reach correct answers through brittle, free-form chains of thought (CoT) that drift, contradict themselves, or hallucinate intermediate steps—failures that compound as reasoning lengthens. Process reward models (PRMs) score intermediate steps but require expensive human step-annotations and struggle to parse unstructured CoT, whose "steps" are not even delimited. We ask: can we make a model's reasoning *intrinsically* easy to supervise, and can a cheap process reward over that structure improve reasoning without destabilizing outcome-based RL?

Method We train the model to reason in a *structured internal Q&A format*: the `<think>` block becomes an alternating sequence of `<ask>` (a purposeful, self-posed question) and `<reflect>` (the computation that answers it) pairs, turning a monologue into discrete, individually-checkable units. We (1) warm-start with supervised fine-tuning (SFT) on structured traces distilled from a teacher LLM, then (2) optimize with RLOO under a composite reward $R = R_{\text{verif}} + \lambda R_{\text{judge}}$, where R_{verif} bundles deterministic checks and R_{judge} is an LLM-judge *process* reward scoring each pair on **relevance** and **quality**. Novelty: self-questioning applied *internally within one trajectory* (vs. prior work that generates external practice problems), gated by a **Correctness Dominance Invariant** ($\lambda < 0.4$) that guarantees any correct answer outscores any incorrect one.

Implementation We build an asynchronous two-worker RLOO trainer (a vLLM sampling worker and a gradient-update worker, orchestrated with Ray on Modal H100s) and run it on Qwen2.5-0.5B for two tasks: **Countdown** (arithmetic equation search) and a **GSM8K** extension. The judge is served by a hosted LLM (Gemini 2.5 Flash-Lite); we sweep $\lambda \in \{0, 0.1, 0.25, 0.35\}$ and evaluate Pass@k with 16 samples/problem.

Results On Countdown, structured Q&A RLOO reaches **62.3%** Pass@1 vs. 48.6% for free-form-CoT RLOO and 28.8% for the SFT base. On GSM8K (full 1,319-problem test set), RLOO improves a structured-SFT baseline from **35.5%/75.2%** to **42.6%/78.5%** Pass@1/Pass@16; the judge adds a small but consistent gain over the λ=0 outcome-only ablation, and the dominance invariant holds throughout (no reward hacking or collapse). We also observe the expected RL *diversity collapse*: Pass@1 rises while entropy and Pass@k gains shrink.

Discussion Enforcing structure makes reasoning legible to an automated judge and lets a cheap, annotation-free process reward fold safely into outcome RL. Gains are large where the search space is wide and the base model weak (Countdown) and modest where the base is already competent and answers short (GSM8K). Limitations: a single 0.5B model, arithmetic-only tasks, a single seed, and a judge that inherits teacher biases.

Conclusion The format of reasoning is a lever for its supervision: structuring reasoning *before* rewarding it is a practical, stable route to step-level supervision, and a promising substrate for distilled PRMs, difficulty-adaptive questioning, and harder domains such as math and code.

Structured Q&A Reasoning for Language Models: Internal Self-Questioning as a Process Reward for RL Fine-Tuning

Anuj Jamwal
anujjam@stanford.edu

Srinidhi Bhat
sri321@stanford.edu

Abstract

Outcome-based RL for reasoning models rewards only the final answer, giving no feedback on intermediate steps and letting free-form chain-of-thought drift and hallucinate. We propose training language models to reason in a *structured internal Q&A format*—alternating self-posed `<ask>` questions and `<reflect>` answers—which decomposes reasoning into discrete, checkable units. We warm-start with SFT on teacher-distilled structured traces, then fine-tune with RLOO under a composite reward that adds an LLM-judge *process* reward R_{judge} (scoring each pair on relevance and quality) to a deterministic verifiable reward R_{verif}, combined as $R_{\text{verif}} + \lambda R_{\text{judge}}$ subject to a Correctness Dominance Invariant ($\lambda < 0.4$) that keeps the outcome signal authoritative. On Countdown, the method reaches 62.3% Pass@1 (vs. 48.6% for free-form RLOO); on a GSM8K extension it improves a structured-SFT baseline from 35.5% to 42.6% Pass@1 (and 75.2% to 78.5% Pass@16) on Qwen2.5-0.5B, with the process reward yielding small consistent gains and no reward hacking. Structuring reasoning makes it cheap and stable to supervise at the step level.

1 Introduction

Large language models solve multi-step reasoning problems by emitting a chain of thought (CoT)—a free-form natural-language monologue—before committing to an answer [Wei et al., 2022]. When such models are improved with reinforcement learning, the reward is almost always *outcome-based*: a binary signal indicating whether the final answer is correct [Guo et al., 2025, Shao et al., 2024]. This paradigm has two well-known weaknesses. First, the reward is *sparse*: a long trajectory collapses to a single bit, so the model receives no guidance about which intermediate steps were sound and which were lucky or wrong. Second, free-form CoT is *unconstrained*: as sequences grow, reasoning can drift, contradict earlier steps, or snowball small errors into confident hallucinations, and a correct final answer can mask an incoherent derivation.

Process reward models (PRMs) address sparsity by scoring intermediate steps [Lightman et al., 2023, Wang et al., 2024], but they are expensive—traditionally requiring large sets of human step-level annotations—and they must operate on messy, free-form traces whose "steps" are not even well delimited. This raises a question that motivates our work: *rather than building an ever-stronger PRM to parse unstructured reasoning, can we change the reasoning format so that it is intrinsically easy to supervise?*

Inspired by how students are taught to reason by explicitly identifying and resolving missing information, we train models to reason in a **structured internal Q&A format**. The standard `<think>` block is replaced by an alternating sequence of two tags: `<ask>`, in which the model poses a purposeful, problem-specific question to itself, and `<reflect>`, in which it computes or reasons to answer that question. This turns a monologue into a sequence of discrete, self-contained reasoning units, each

checkable independently—by deterministic rules *and* by a lightweight LLM judge. We then fold a cheap, annotation-free process reward over this structure into standard RLOO [Ahmadian et al., 2024] fine-tuning, under a safety constraint that prevents the process reward from ever overriding correctness.

Research questions and hypotheses. We investigate three questions:

RQ1. Does reasoning in a structured `<ask>`/`<reflect>` format help an RL-trained reasoner relative to free-form CoT? *(H1: yes—structure both regularizes reasoning and exposes more reward surface.)*

RQ2. Does adding an LLM-judge *process* reward improve reasoning beyond an outcome-only (λ=0) baseline, *without* reward hacking or training collapse? *(H2: a small judge weight λ gives consistent gains while the dominance invariant preserves stability.)*

RQ3. How does the benefit depend on the task? *(H3: gains are largest when the search space is wide and the base model weak, e.g. Countdown, and smaller when answers are short and the base model competent, e.g. GSM8K.)*

We answer these on two arithmetic-reasoning benchmarks, Countdown and GSM8K [Cobbe et al., 2021], using Qwen2.5-0.5B [Yang et al., 2024], and we contribute: (i) a structured self-questioning reasoning format and a teacher-distillation SFT pipeline; (ii) a composite verifiable-plus-judge reward with a provable Correctness Dominance Invariant; (iii) an asynchronous two-worker RLOO implementation; and (iv) an empirical study—λ ablation, training-dynamics analysis, and qualitative failure analysis—of when internal self-questioning helps.

2 Related Work

RL for reasoning. Outcome-reward RL has become the dominant recipe for eliciting reasoning, from RLHF/PPO [Schulman et al., 2017, Ouyang et al., 2022] to value-free policy-gradient variants tailored to LLMs such as RLOO [Ahmadian et al., 2024] and GRPO [Shao et al., 2024], and most visibly in DeepSeek-R1 [Guo et al., 2025], which shows pure outcome RL can induce long CoT. We build directly on RLOO, whose leave-one-out baseline is simple and well-suited to verifiable-reward tasks, and we keep its outcome signal authoritative while augmenting it with a structured process reward.

Process supervision and PRMs. Lightman et al. [2023] show process supervision outperforms outcome supervision on math, but rely on a large human-annotated step dataset; Math-Shepherd [Wang et al., 2024] reduces this cost via automatic per-step value estimation. A core difficulty for all PRMs is that free-form CoT has no explicit step boundaries. Our work attacks this at the source: by emitting reasoning as discrete `<ask>`/`<reflect>` pairs, each "step" is delimited and individually scorable, so an off-the-shelf LLM judge [Zheng et al., 2023] can act as a cheap, annotation-free PRM.

Structured and self-questioning reasoning. Prompting methods impose structure at inference time—self-consistency [Wang et al., 2023], Tree-of-Thoughts [Yao et al., 2023], Self-Refine [Madaan et al., 2023], and Self-Ask [Press et al., 2022], which decomposes a question into follow-up sub-questions—but they do not *train* the policy to internalize the structure, nor reward step quality. Most related, Self-Questioning Language Models (SQLM) [Chen et al., 2025] use self-play in which a proposer generates *external* practice problems for a solver. Our use of self-questioning differs in locus and purpose: it is *internal* to a single problem's trajectory, constraining how the model reasons *about the problem at hand* rather than what problems it practices on. Our structured SFT data is distilled from cognitive-behavior reasoning traces in the spirit of Gandhi et al. [2025] and Gandhi et al. [2024].

Positioning. Relative to PRMs we remove the need for human step-labels by making steps machine-checkable; relative to inference-time structuring we bake the structure into the policy via SFT+RL and explicitly reward step quality; relative to external self-questioning we apply it internally as a reasoning constraint. To our knowledge, combining an *internal* self-questioning format with an LLM-judge process reward, gated by a correctness-dominance constraint inside RLOO, is novel.

3 Method

Our method has three components: (i) a structured Q&A reasoning format, (ii) an SFT warm-start that teaches the format, and (iii) RLOO fine-tuning under a composite verifiable-plus-judge reward with a correctness-dominance constraint. The pipeline proceeds in three stages: *teacher-distilled structured SFT data → SFT policy → RLOO with reward* $R_{\text{verif}} + \lambda R_{\text{judge}}$ (Algorithm 1).

3.1 Structured Q&A reasoning format

A model response is required to follow the schema

$$\texttt{<think>}\ \big(\texttt{<ask>}q_i\texttt{</ask>}\ \texttt{<reflect>}r_i\texttt{</reflect>}\big)_{i=1}^{N}\ \texttt{</think>}\ \texttt{<answer>}a\texttt{</answer>},$$

where each q_i is a purposeful question (it must end with "?") that makes a concrete observation, proposes a hypothesis, or narrows the solution space, and each r_i is the computation or insight that answers q_i. The final pair is expected to *verify* the solution. This decomposition is the key enabler: it turns reasoning into N discrete units with explicit boundaries, so both deterministic rules and an LLM judge can score each unit in isolation.

3.2 SFT warm-start via teacher distillation

A pretrained 0.5B model rarely produces well-formed <ask>/<reflect> traces zero-shot, so we first teach the format by SFT. We construct a synthetic dataset by *rewriting* existing reasoning traces into the structured format with a strong teacher LLM (Gemini 3.1 Pro), instructed to (1) make every <ask> a purposeful, problem-grounded question (no generic filler such as "How should I approach this?"), (2) place a <reflect> immediately after each <ask>, (3) preserve *all* computations from the original trace, (4) emit no free text outside the tags, (5) end with a verification pair, and (6) preserve the final answer exactly. For Countdown we rewrite the cognitive-behavior traces of `Asap7772/cog_behav_all_strategies` [Gandhi et al., 2025]; for GSM8K we rewrite `openai/gsm8k` solutions. We then SFT Qwen2.5-0.5B on these traces with a *response-only* (assistant-token) cross-entropy loss: prompt tokens are masked so the loss is averaged only over completion tokens. Hyperparameters are in Table 1.

3.3 Composite reward design

For a trajectory τ we define the total reward

$$R(\tau) \;=\; R_{\text{verif}}(\tau) \;+\; \lambda\, R_{\text{judge}}(\tau). \tag{1}$$

Verifiable reward. R_{verif} is a sum of four deterministic terms computed without any model call:

$$R_{\text{verif}} = \underbrace{R_{\text{correct}}}_{\{0,1\}} + \underbrace{R_{\text{format}}}_{\{0,0.1\}} + \underbrace{R_{\text{length}}}_{[-0.1,0]} + \underbrace{R_{\text{QA}}}_{[-0.1,0]} \;\in\; [-0.2,\, 1.1]. \tag{2}$$

R_{correct} checks the extracted <answer> against ground truth (exact equation evaluation for Countdown; numeric/string match for GSM8K). R_{format} awards 0.1 iff the trace contains a <think> block, an <answer> block, and ≥ 1 valid <ask>/<reflect> pair. R_{length} applies a linear penalty (capped at -0.1) for more than $n_{\max}$=5 pairs. R_{QA} applies -0.02 per structural violation (an <ask> not ending in "?", or an orphaned tag), capped at -0.1.

Judge (process) reward. R_{judge} scores reasoning *quality* with a single LLM-judge call per trajectory. The judge receives the problem and the <think> block and emits, for each pair i, two integer scores in $\{-1, 0, +1\}$: **Relevance** ρ_i (-1 if the <ask> is not a real question or is trivial/redundant; 0 if relevant but generic; $+1$ if purposeful and problem-specific) and **Quality** κ_i (-1 if the <reflect> contains a computation error, hallucination, or contradiction; 0 if correct but shallow; $+1$ if correct, insightful, and logically entailed). The per-trajectory reward averages over the N pairs:

$$R_{\text{judge}}(\tau) = \frac{1}{N}\sum_{i=1}^{N} \frac{\rho_i + \kappa_i}{2} \;\in\; [-1,\, 1]. \tag{3}$$

The judge is one inference per trace (all traces in a batch judged in parallel), making this an *annotation-free* PRM. Traces with no valid pairs receive R_{judge}=0 (already penalized by R_{format}).

Algorithm 1 Structured Q&A RLOO with verifiable + judge reward

Require: SFT policy π_θ, reference π_{ref}, prompts $\mathcal{D}$, group size G, judge weight λ, judge LLM J

for step $= 1 \dots T$ **do**
  Sample prompts $\{x_b\}$; for each x_b sample G trajectories $\{\tau_{b,g}\} \sim \pi_\theta(\cdot \mid x_b)$ ▷ vLLM worker
  $R_{\text{verif}}(\tau) \leftarrow R_{\text{correct}} + R_{\text{format}} + R_{\text{length}} + R_{\text{QA}}$ for each τ ▷ deterministic
  $\{R_{\text{judge}}(\tau)\} \leftarrow J(\{\tau\})$ ▷ one parallel batch of judge calls
  $R(\tau) \leftarrow R_{\text{verif}}(\tau) + \lambda\, R_{\text{judge}}(\tau)$ ▷ invariant guarantees $\lambda < 0.4$
  Compute leave-one-out advantages $A_{b,g}$ within each group of G
  Update θ on $\mathcal{L}(\theta)$ via gradient accumulation; periodically sync θ to the sampler ▷ update worker
end for

Correctness Dominance Invariant. A process reward must not let a beautifully-reasoned *wrong* answer outscore an ugly *correct* one. We choose λ to guarantee this. The worst-scoring *correct* trace earns $R_{\text{verif}} \geq 0.9$ (correctness 1.0, valid format $+0.1$, both penalties maxed at -0.2), so its total is $\geq 0.9 - \lambda$. The best-scoring *incorrect* trace earns $R_{\text{verif}} \leq 0.1$, so its total is $\leq 0.1 + \lambda$. Requiring $0.9 - \lambda > 0.1 + \lambda \Leftrightarrow \boxed{\lambda < 0.4}$ ensures every correct answer outranks every incorrect one regardless of reasoning quality; the process reward can only *re-rank within* the correct and incorrect groups. We sweep $\lambda \in \{0, 0.1, 0.25, 0.35\}$, all safely inside the invariant.

3.4 RLOO fine-tuning

We optimize the SFT policy with RLOO [Ahmadian et al., 2024]. For each prompt we sample a group of G trajectories and compute the leave-one-out advantage $A_i = R_i - \frac{1}{G-1}\sum_{j \neq i} R_j$, i.e. each sample is baselined by the mean reward of its group-mates. The loss (with an optional importance ratio $r_i(\theta) = \pi_\theta(\tau_i)/\pi_{\text{old}}(\tau_i)$ clamped to ≤ 10), an entropy bonus, and a KL penalty to the frozen SFT reference are

$$\mathcal{L}(\theta) = -\,\mathbb{E}_i\big[\, r_i(\theta)\, A_i \,\big] \;-\; \beta_H\, \mathcal{H}[\pi_\theta] \;+\; \beta_{\text{KL}}\, \text{KL}\big(\pi_\theta \,\|\, \pi_{\text{ref}}\big). \tag{4}$$

System architecture. Sampling and optimization run as two Ray actors on separate GPUs, orchestrated on Modal. A **sampling worker** hosts a vLLM engine and generates G completions per prompt (stop token `</answer>`); an **update worker** holds the policy, reference model, optimizer, and scheduler, with gradient checkpointing and accumulation. The orchestrator alternates sample→score→update and re-syncs checkpoints to the sampler. The judge is served via a hosted LLM API with concurrent requests, so judging overlaps with computation.

4 Experimental Setup

Tasks and datasets. **Countdown**: given a target and a multiset of numbers, produce an arithmetic equation using each number once that evaluates to the target; correctness is verified symbolically. **GSM8K** [Cobbe et al., 2021]: grade-school math word problems with a numeric answer; we evaluate on the full test split (1,319 problems). Both are *verifiable*, ideal for studying outcome-vs-process reward trade-offs.

Base model. All experiments use `Qwen/Qwen2.5-0.5B` [Yang et al., 2024] (the Countdown Pass@k scaling study additionally uses Qwen2.5-3B). RLOO is initialized from the task's structured-SFT checkpoint and uses that same checkpoint as the frozen KL reference.

Baselines. (i) *SFT (free-form CoT)* and *SFT (structured Q&A)* measure the format's effect before RL. (ii) *RLOO (free-form CoT)* isolates the value of structure under identical RL. (iii) Within structured RLOO, λ=0 is an outcome-only ablation and $\lambda \in \{0.1, 0.25, 0.35\}$ test the process reward. (iv) An *IPO* [Azar et al., 2024] preference-optimized model is a non-RLOO Countdown reference. The free-form-vs-structured contrast (i–ii) is measured on Countdown; for GSM8K we report the full ladder—*zero-shot base*, structured SFT, and RLOO—leaving a free-form-CoT RLOO run on GSM8K to future work. These let us attribute gains to *structure* (RQ1, Countdown) and to the *judge reward* (RQ2, both tasks).

Table 1: Training and evaluation configuration (GSM8K extension; Countdown RLOO uses batch size 16, otherwise analogous).

Stage	Hyperparameter	Value
SFT	base model	Qwen2.5-0.5B
	learning rate / schedule	5×10^{-5} / cosine, warmup 0.05
	epochs / batch size / grad accum	6 / 64 / 8
	max prompt / response length	512 / 1024
	optimizer / weight decay / clip	AdamW / 0.01 / 1.0
	loss	response-token (assistant-only) CE
RLOO	policy / reference init	structured-SFT checkpoint
	learning rate / schedule	1×10^{-5} / constant, no warmup
	batch size / group size G	128 / 8
	grad accum / training steps	128 / 100
	KL coeff β_{KL} / entropy coeff β_H	0.001 / 0.001
	rollout temp / top-p / top-k	1.0 / 1.0 / -1
	max model / response length	2048 / 1024
	judge model / concurrency	Gemini 2.5 Flash-Lite / 20
	judge weight λ (sweep)	$\{0,\ 0.1,\ 0.25,\ 0.35\}$
Eval	samples n / temp / top-p / top-k	16 / 0.6 / 0.95 / 20
	test set	GSM8K test (1,319 problems)
		Countdown Eval (50 problems)
Infra	hardware / framework	Modal H100 / Ray + vLLM

Table 2: Countdown Pass@1 (Qwen2.5-0.5B, 16 samples/problem). Structure and the judge reward each help; RLOO more than doubles the SFT base.

Configuration	Pass@1
SFT (free-form CoT)	28.75%
SFT (structured Q&A)	30.40%
RLOO (free-form CoT)	48.63%
RLOO (structured, λ=0)	59.90%
RLOO (structured, λ=0.1)	61.80%
RLOO (structured, λ=0.25)	60.10%
RLOO (structured, λ=0.35)	**62.30%**

Metrics. We report **Pass@k**: for each problem we draw n=16 samples and use the unbiased estimator of the probability that at least one of k is correct [Cobbe et al., 2021]. Pass@1 measures single-shot reliability; larger k measures whether a correct solution is *reachable* (coverage/diversity). We also track training-set rollout accuracy and policy entropy from the RL logs. Evaluation uses temperature 0.6, top-p 0.95, top-k 20, max 1024 new tokens.

5 Results

5.1 Quantitative Evaluation

Countdown. Table 2 reports Countdown Pass@1. **Structure helps (RQ1):** structured Q&A beats free-form CoT both before RL (30.4 vs. 28.8) and, dramatically, after RL ($\geq$ 59.9 vs. 48.6)—a $+$11-point Pass@1 gain from structure alone under identical RLOO. **The judge helps (RQ2):** every λ>0 run improves on the λ=0 ablation, with λ=0.35 best at 62.3%; RL more than doubles the SFT base (30.4 $\rightarrow$ 62.3). Figure 1 plots Pass@k for the 0.5B model (judge weights track closely at low k; lower λ gives best high-k coverage), and Figure 2 repeats the study on Qwen2.5-3B: the RL variants dominate at low k (Pass@1 $\approx$ 55–61% vs. 40%) but the curves *cross*—by k=16 the SFT model matches or exceeds them, the signature of RL *diversity collapse* (Figure 4).

GSM8K. Table 3 and Figure 3 report GSM8K Pass@k for the structured-SFT baseline and the RLOO λ sweep on the full test set. **RL clearly helps:** RLOO lifts the structured-SFT baseline

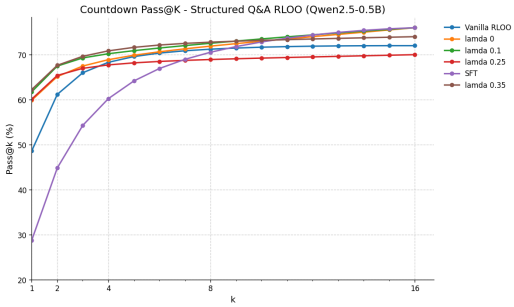

Figure 1: Countdown Pass@k (Qwen2.5-0.5B, 50-problem eval) across judge weights λ. Higher λ leads at small k; lower λ gives the best high-k coverage.

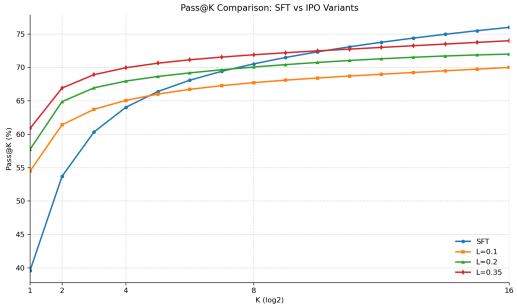

Figure 2: Countdown Pass@k on Qwen2.5-3B: structured RLOO (λ) variants vs. SFT. RL boosts low-k accuracy but SFT catches up at high k (diversity collapse).

Table 3: GSM8K Pass@k (%): zero-shot base, structured-SFT baseline, and RLOO across judge weights λ (Qwen2.5-0.5B, 1,319 test problems, 16 samples). **Bold** = best per column; RLOO improves on the SFT baseline at every k. †The base model emits free-form CoT, not the trained `<answer>` schema, and is scored by numeric extraction from its (verbose, repetitive) output; this is *approximate*—Pass@1 spans ≈12–19% across reasonable extractors (one fixed extractor shown), and 0% under the exact `<answer>` scorer used for the trained rows.

Configuration	Pass@1	Pass@2	Pass@4	Pass@8	Pass@16
Base (zero-shot, free-form CoT)†	15.20	25.39	38.42	52.48	65.43
SFT (structured, no RL)	35.54	46.65	57.44	67.07	75.21
RLOO (λ=0)	41.50	52.19	61.70	69.92	76.95
RLOO (λ=0.1)	40.81	51.74	61.86	70.82	**78.54**
RLOO (λ=0.25)	42.11	52.71	62.37	70.70	77.79
RLOO (λ=0.35)	**42.57**	**53.11**	**62.69**	**71.03**	78.17

from 35.5% → 42.6% Pass@1 (+7.0 points, best at λ=0.35) and 75.2% → 78.5% Pass@16, improving on SFT at *every* k; the margin is largest at small k (+7.0 at Pass@1 vs. +3.3 at Pass@16), so RL improves single-shot reliability most. For reference, the *zero-shot* base model already produces free-form CoT and reaches roughly 15% Pass@1 / 65% Pass@16 when scored by numeric extraction—though only approximate, since its verbose output makes extraction ambiguous (≈12–19% Pass@1 across reasonable extractors; 0% under the structured `<answer>` scorer it never emits). The base→SFT→RLOO ladder (≈15→35.5→42.6% Pass@1) shows large cumulative gains, but the base→SFT step bundles training *and* the format switch, so it is not a clean structure-vs-free-form isolation—that contrast we draw only on Countdown. **The judge gives a small additional gain (RQ2):** λ=0.35 is best at Pass@1–Pass@8 (Pass@1 42.57 vs. 41.50 for λ=0), while λ=0.1 edges out the field at Pass@16. The differences across λ are modest ($\leq$ 2 Pass@1 points)—consistent with H3: GSM8K answers are short, the SFT base already competent, and the verifiable signal dense, leaving less surface for the judge than in Countdown's wide search space. No λ caused reward hacking or collapse, empirically confirming the dominance invariant (§3.3).

Training dynamics. Figures 4 (Countdown) and 5 (GSM8K) track the 100 RLOO steps and reveal two phenomena at different magnitudes. *(a) Learning.* Train rollout accuracy (sampling temperature 1.0) rises steeply on Countdown—∼8–18% to ∼47–48%, a 3–5× gain—vs. a gentler ∼36–43% to ∼50–53% on GSM8K, matching H3 (Countdown is search-heavy with a weak SFT base; GSM8K's base is already competent). Judge-augmented runs are among the strongest at convergence (GSM8K λ=0.25/0.35 reach 51.5%/53.2%). *(b) Diversity collapse.* Policy entropy falls monotonically for every run, far more sharply on Countdown (∼0.78→0.34–0.54 nats) than GSM8K (∼0.21→0.15). This entropy decay is the mechanism behind the Pass@k crossover in Figure 2: RL concentrates probability on a few high-reward trajectories, raising Pass@1 but eroding the diversity that drives

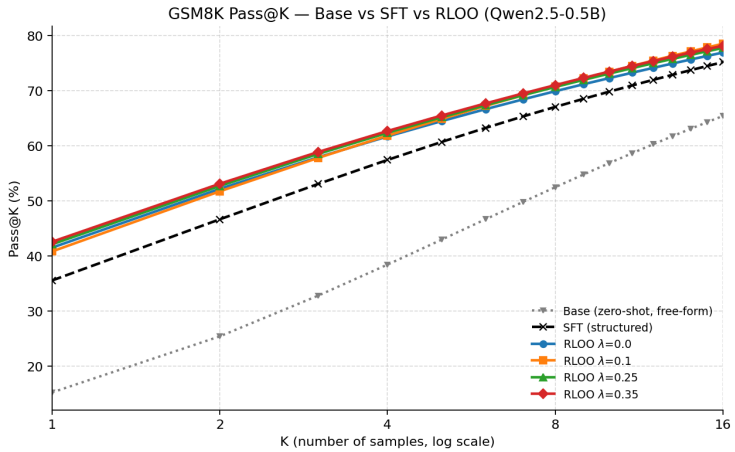

Figure 3: GSM8K Pass@k: zero-shot base (dotted; free-form CoT, approximate—see Table 3 note), structured SFT (dashed), and RLOO at four judge weights. RLOO improves on SFT at every k, with the largest margin at small k; among RLOO runs higher λ (0.25–0.35) is marginally best at small k and λ=0.1 at k=16.

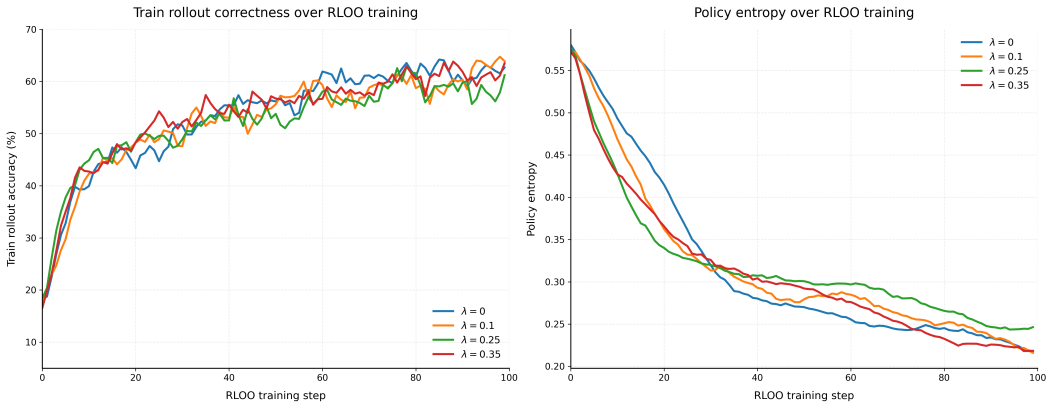

Figure 4: Countdown RLOO training dynamics over 100 steps (Qwen2.5-0.5B). (a) Rollout accuracy rises $\sim$3.5–4$\times$ (from $\sim$17% to over 60%); (b) policy entropy collapses from $\sim$0.58 to 0.21–0.25 nats—a far steeper diversity collapse than GSM8K.

high-k coverage. The largest judge weight collapses entropy the most on Countdown, so practitioners trading single-shot accuracy for coverage should prefer smaller λ.

5.2 Qualitative Analysis

Successful structured reasoning. The model learns to emit clean, problem-grounded `<ask>`/`<reflect>` chains that decompose the problem and self-verify. A representative correct GSM8K trace (target 18):

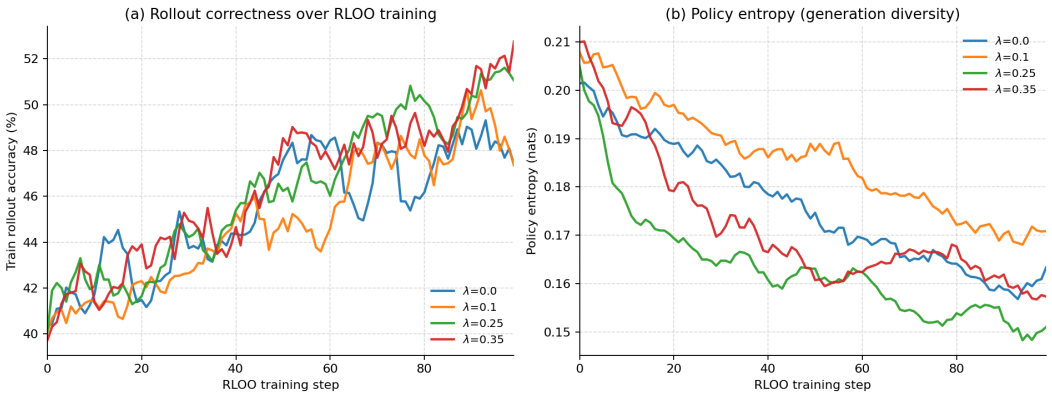

Figure 5: GSM8K RLOO training dynamics over 100 steps. (a) Train rollout accuracy improves for all λ, with λ=0.25/0.35 highest. (b) Policy entropy decays for all runs (diversity collapse), explaining the diminishing Pass@k gains at large k.

Problem. Janet's ducks lay 16 eggs/day; she eats 3 for breakfast and bakes with 4; she sells the rest at \$2/egg. How much does she make daily?
<ask> *How many eggs does Janet consume in a day (breakfast and baking)?* </ask> <reflect> $3 + 4 = 7$ *eggs.* </reflect>
<ask> *How many eggs remain for sale?* </ask> <reflect> $16 - 7 = 9$ *eggs.* </reflect>
<ask> *How much money does selling 9 eggs at \$2 yield?* </ask> <reflect> $9 \times 2 = 18$. </reflect>
<ask> *Does* $16 - 3 - 4$ *then* $\times 2$ *equal 18?* </ask> <reflect> $16 - 3 = 13$, $13 - 4 = 9$, $9 \times 2 = 18$. *Consistent.* </reflect>
Answer. 18 *(judge: relevance/quality all* $+1$*; final pair is a genuine verification)*

The questions are specific (not filler), each reflection performs one computation, and the final pair verifies—exactly the behavior the judge's relevance/quality axes reward.

Failure modes. Errors are overwhelmingly *semantic, not structural*: the format is well-formed but a reflection misreads the problem or makes an arithmetic slip. A characteristic failure (target 70000):

Problem. Josh buys a house for \$80,000, puts \$50,000 into repairs, which increases the value by 150%. What profit did he make?
<ask> *What is the total cost after repairs?* </ask> <reflect> $80{,}000 \times 1.5 = 120{,}000$. </reflect> ← misreads "+150%", ignores the \$50k cost
<ask> *What is the net change vs. the original value?* </ask> <reflect> $120{,}000 - 80{,}000 = 40{,}000$. </reflect>
<ask> *How much profit results?* </ask> <reflect> $40{,}000 \times 1.5 = 60{,}000$. </reflect> ← compounds the error
Answer. 60000 *(wrong; correct: new value* $= 80000 \times 2.5 = 200000$*, profit* $= 70000$*)*

Here the structure is perfect but the *quality* axis should fire (-1): the model misinterprets "increased by 150%". This is precisely the signal R_{judge} targets, yet at 0.5B scale the policy still makes such conceptual errors, capping how much a process reward can help. Across the λ=0.35 run, 201/1319 problems are solved on all 16 samples while 288/1319 are never solved—a hard tail requiring multi-step setup or world knowledge that structure alone does not fix.

Why the method works (and its ceiling). Structure helps most when the bottleneck is *search* (Countdown), where decomposing into hypothesis-test pairs and self-verifying prunes invalid equations—hence the large Pass@1 gains. On GSM8K the bottleneck is often *comprehension* (correctly setting up the arithmetic), which structure organizes but does not solve; combined with a strong dense verifiable signal, this leaves the judge little headroom, explaining the modest λ effect.

6 Discussion

Limitations. (1) *Scale and domain:* a single 0.5B model on arithmetic reasoning; gains may differ at larger scale or on open-ended tasks. (2) *Judge fidelity and cost:* R_{judge} is only as reliable as the judge LLM, which can be miscalibrated or style-biased, and adds an API call per trace. (3) *Small-set Countdown evaluation:* Countdown numbers come from a 50-problem eval (indicative); GSM8K uses the full 1,319-problem test set. (4) *Single seed:* runs are not yet repeated, so small GSM8K λ differences are within plausible noise. (5) *Distillation ceiling:* structured SFT data inherits the teacher's reasoning.

Broader impact. Making reasoning legible to automated judges lowers the cost of process supervision and improves *auditability* of model reasoning—useful for oversight. The same legibility could be gamed (a model producing convincing-looking traces); our dominance invariant mitigates this for verifiable tasks but not domains lacking ground truth.

Difficulties encountered. The main challenges were (i) keeping the asynchronous sampler/updater in sync without stale-policy drift, (ii) tolerating judge-API latency and malformed JSON (handled with parallel calls, retries, and neutral-on-failure scoring), and (iii) tuning λ to stay inside the dominance invariant—an early over-weighted judge encouraged verbose, low-content pairs until the length/QA penalties and the invariant bound were added.

7 Conclusion

We showed that *structuring reasoning before rewarding it* is a practical, stable route to step-level supervision. Training models to reason in an internal `<ask>`/`<reflect>` format makes intermediate steps discrete and machine-checkable, letting a cheap, annotation-free LLM-judge process reward fold into outcome-based RLOO—kept safe by a Correctness Dominance Invariant ($\lambda < 0.4$). Structure yields large gains where reasoning is search-heavy (Countdown: $28.8 \rightarrow 62.3\%$ Pass@1) and the judge adds small consistent gains; on GSM8K it improves a $35.5/75.2\%$ structured-SFT baseline to $42.6/78.5\%$ Pass@1/Pass@16 with stable training and no reward hacking. The take-home: *the format of reasoning is a lever for its supervision*. Next steps: (1) multiple seeds and a free-form-CoT RLOO baseline on GSM8K (the structured-SFT baseline is now included); (2) using the structured judge to train a distilled, inference-cheap PRM; (3) *dynamic* questioning depth conditioned on difficulty; and (4) transfer to code generation, where unit tests give a natural verifiable reward.

8 Team Contributions

Both authors contributed substantially and equitably; the split below reflects primary ownership.

- **Anuj Jamwal:** RLOO trainer (two-worker Ray/vLLM/Modal architecture, leave-one-out advantage, update worker), reward wiring and the Correctness Dominance Invariant, the λ sweep, evaluation pipeline and Pass@k analysis, training-dynamics figures.
- **Srinidhi Bhat:** structured-Q&A SFT pipeline, IPO Trainer and teacher-distillation dataset generation (Countdown and GSM8K) with prompt engineering, the LLM-judge design (relevance/quality rubric and prompts), qualitative/failure analysis, and the report and related-work survey.

Changes from Proposal As planned in the proposal/poster, we completed the λ sweep over the dominance penalty and extended the method from Countdown to GSM8K, evaluating on the full GSM8K test set and adding the zero-shot-base and structured-SFT baselines. The main deviation is that multi-seed variance estimates remain in progress; we report single-seed results and flag the affected comparisons as indicative.

9 AI Tools Disclosure

In accordance with the course academic integrity guidelines, we disclose the use of generative AI tools as coding and writing aids. The split between AI-aided and independent work is detailed below:

- **AI-Assisted Components:** AI tools were used strictly as an aid for boilerplate generation, data pipelines, and writing support:
  - *Data Pipelines (Extension):* Drafting SFT dataset generation scripts that make parallel asynchronous API calls to rewrite raw traces into QA format (`generate_gsm8k_sft.py` and `generate_sft_dataset.py`).
  - *Boilerplate (Extension):* Writing utility code for parsing, JSON handling, and formatting inside the LLM-judge process reward model (`llm_judge.py`).
  - *Writing Assistance:* Proofreading, editing, and formatting drafts of this final report to improve clarity and LaTeX layout.
- **Independently Developed Components:** All essential project components and core reinforcement learning logic were developed entirely independently without AI tools:
  - *SFT Loop:* The implementation of response-token masking and forward/backward training logic in `sft.py`.
  - *IPO Objective:* The policy/reference log-probability computations and pairwise preference loss optimization in `ipo.py`.
  - *RLOO Algorithm:* The complete implementation of the reinforcement learning update worker, including rollout advantage computations, leave-one-out baselining, policy gradient updates, and entropy/KL regularization in `rloo_update_worker.py`.
  - *Evaluation:* The execution of Pass@k metrics, evaluations on the test sets, and final training-dynamics analysis.

References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2024.

L. Chen et al. Self-questioning language models. *arXiv preprint arXiv:2508.03682*, 2025.

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

Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D. Goodman. Stream of search (sos): Learning to search in language. In *Conference on Language Modeling (COLM)*, 2024.

Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D. Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *arXiv preprint arXiv:2503.01307*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

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

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*, 2022.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

An Yang, Baosong Yang, Beichen Zhang, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

A Implementation and Reproducibility Details

Artifacts. Structured SFT datasets: `dragonite321/cog_behav_all_strategies_structured_qa` (Countdown, rewritten from `Asap7772/cog_behav_all_strategies`) and `dragonite321/gsm8k_structured_qa` (rewritten from `openai/gsm8k`). The codebase is available at `https://github.com/srinidhi321/cs224r_default_proj`. Policy/reference base: `Qwen/Qwen2.5-0.5B`. RLOO is implemented as two Ray actors (a vLLM *sampling worker* and a gradient *update worker*) on Modal H100 GPUs; the judge is served via a hosted LLM API (Gemini 2.5 Flash-Lite) with concurrency 20, one inference per trace.

Evaluation. Pass@k uses the unbiased estimator over $n{=}16$ samples (temperature 0.6, top-p 0.95, top-k 20, 1024 max new tokens) on the full GSM8K test split (1,319 problems). The trained models (SFT, RLOO) emit the `<answer>` schema and are scored by exact tag extraction. The zero-shot base model produces free-form CoT and is scored by a numeric-extraction heuristic over the saved generations (a number is recovered in 98.3% of samples); because the base output is verbose and repetitive, this score is approximate (Pass@1 spans $\approx$12–19% across reasonable extractors), and we report one fixed extractor in Table 3.