One Refiner to Unlock Them All: Inference-Time Reasoning Elicitation via Reinforcement Query Refinement

Yixiao Zhou[1,2], Dongzhou Cheng[2,3], Zhiliang Wu[1], Yi Yang[1], Yu Cheng[2,4], Hehe Fan[1]

[1]Zhejiang University, [2]Shanghai Innovation Institute, [3]Southeast University, [4]The Chinese University of Hong Kong
{12421181,wu_zhiliang,yangyics,hehefan}@zju.edu.cn, 230249457@seu.edu.cn, chengyu@cse.cuhk.edu.hk

Correspondence: chengyu@cse.cuhk.edu.hk, hehefan@zju.edu.cn

Abstract

Large Language Models (LLMs) often fail to utilize their latent reasoning capabilities due to a distributional mismatch between ambiguous human inquiries and the structured logic required for machine activation. Existing alignment methods either incur prohibitive $O(N)$ costs by fine-tuning each model individually or rely on static prompts that fail to resolve query-level structural complexity. In this paper, we propose ReQueR (**Re**inforcement **Que**ry **R**efinement), a modular framework that treats reasoning elicitation as an inference-time alignment task. We train a specialized Refiner policy via Reinforcement Learning to rewrite raw queries into explicit logical decompositions, treating frozen LLMs as the environment. Rooted in the classical Zone of Proximal Development from educational psychology, we introduce the Adaptive Solver Hierarchy, a curriculum mechanism that stabilizes training by dynamically aligning environmental difficulty with the Refiner's evolving competence. ReQueR yields consistent absolute gains of 1.7%–7.2% across diverse architectures and benchmarks, outperforming strong baselines by 2.1% on average. Crucially, it provides a promising paradigm for one-to-many inference-time reasoning elicitation, enabling a single Refiner trained on a small set of models to effectively unlock reasoning in diverse unseen models. Code is available at https://github.com/newera-xiao/ReQueR.

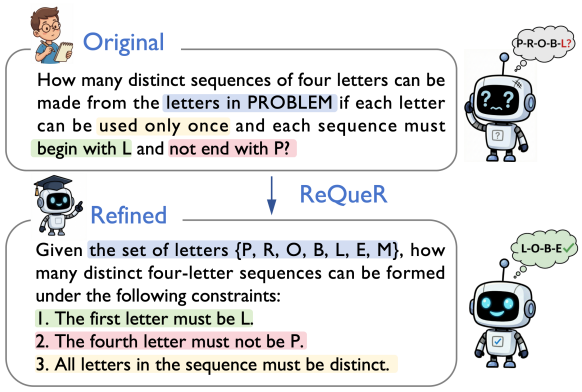

Figure 1: **Eliciting latent reasoning via inference-time query refinement.** A Solver struggles with the ambiguity of a raw human query (top). ReQueR functions as an optimization frontend, reformulating the input into a structured, explicit format (bottom). By resolving ambiguity and employing strategies like variable-mapping, the Refiner aligns the query with the Solver's reasoning patterns, effectively unlocking its latent capabilities without requiring parameter updates.

1 Introduction

Large Language Models (LLMs) possess significant reasoning potential (Shao et al., 2024), yet a gap remains between their intrinsic knowledge and actual performance. Their reasoning capabilities are often not fully activated and are highly sensitive to input phrasing, where small perturbations can lead to failure (Zhao et al., 2021; Mirzadeh et al., 2024; Cai et al., 2025; An et al., 2025). We argue that this bottleneck stems from a distributional mismatch: while human inquiries are often ambiguous and noisy, LLM reasoning engines require explicit and structured logic to function optimally (Deng et al., 2023). Therefore, the key to unlocking latent reasoning lies in bridging this alignment gap between human intent and machine activation.

Bridging this gap, however, is far from trivial. First, parameter-update methods, such as Supervised Fine-Tuning (SFT) (Hu et al., 2022; Xiao et al., 2026b; Chang et al., 2026; Dong et al., 2025) and Reinforcement Learning from Verifiable Rewards (RLVR) (Guo et al., 2025), attempt to align models by modifying internal weights. While effective, these approaches are impractical for agent systems where models are frequently swapped or upgraded. Aligning N models necessitates N independent training runs, leading to a $O(N)$ complexity. Furthermore, they are inapplicable to proprietary black-box models where gradient access

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 38957–38978
July 2-7, 2026

is restricted. Alternatively, while prompt engineering offers a lightweight substitute, most existing strategies act as static wrappers that merely append shallow instructions to the input (Wei et al., 2022). They often fail to perform the deep rewriting required to simplify complex query structures. Moreover, automated prompt optimization tends to be overfitted to specific models or datasets, severely limiting their generalization to unseen environments (Yang et al., 2023). Consequently, existing paradigms fail to provide a universal solution at inference time, as the optimal reasoning trigger remains highly task-dependent and varies significantly across divergent model architectures.

To address these gaps, we introduce ReQueR (**Re**inforcement **Que**ry **R**efinement), an inference-time frontend that reformulates human queries to activate the reasoning capabilities of downstream LLMs. Instead of per-model fine-tuning or task-level prompt optimization, we use reinforcement learning to train a Refiner LLM as a unified meta-policy. As illustrated in Figure 1, ReQueR performs query-level reformulation, translating ambiguous human intent into structured machine logic. During training, the Refiner is optimized through reward signals from frozen Solver LLMs responding to the reformulated queries. This approach shifts the alignment burden from the $O(N)$ cost of independent model-tuning to a scalable $O(1)$ inference-time paradigm, while offering a granularity that static prompt optimization lacks. Once trained, a single Refiner can unlock the reasoning potential of diverse, unseen models without any additional parameter updates. Unlike dataset-level prompt optimization, ReQueR reformulates each query individually, ensuring that the reasoning trigger is tailored to the specific logic of every sample.

However, training this Refiner policy presents significant challenges. The main problem is reward sparsity (Yu et al., 2025; Ding et al., 2026; Li et al., 2025a; Yu et al., 2026; Li et al., 2026) caused by the use of static training environments. Solvers that are either too weak or overly capable fail to provide informative feedback, often driving the Refiner to exploit model-specific biases rather than learning general reasoning logic. Another problem is reward hacking, where the Refiner learns to cheat by directly leaking the answer into the refined query (Gao et al., 2023). While this results in high training scores, it leads to catastrophic failure when the Refiner is applied to new, unseen environments.

To address these challenges, we introduce two mechanisms. First, the Adaptive Solver Hierarchy (ASH) mitigates reward sparsity and model-specific overfitting. Grounded in the Zone of Proximal Development (Vygotsky, 1978), ASH utilizes a pool of Solvers of varied capabilities and performs per-sample dynamic matching according to Refiner's current competence. By escalating Solver difficulty upon consistent failure and de-escalating when success becomes trivial, ASH ensures the Refiner evolves within its optimal growth zone, maintaining a dense reward signal (Bengio et al., 2009; Xiao et al., 2026a; Li et al., 2025b) without model-specific dependencies. Second, we implement a perplexity-based constraint to suppress reward hacking. Unlike computationally expensive and unstable generative LLM judges (Wang et al., 2024a), our approach uses a single forward pass to detect potential leakage, enabling quick monitoring within the RL loop.

Our contributions can be summarized as follows:

- We propose **ReQueR**, a modular inference-time Query Refinement framework. By learning a unified meta-policy for per-sample query refinement, ReQueR unlock the reasoning potential of diverse LLMs.
- We introduce the Adaptive Solver Hierarchy alongside an efficient leakage detection constraint. ASH addresses reward sparsity through per-sample dynamic Solver selection tailored to the Refiner's refined queries, while our perplexity-based constraint effectively mitigates reward hacking.
- Extensive experiments across various LLMs and reasoning tasks demonstrate that ReQueR consistently improves reasoning performance, and outperforms existing baselines. Notably, we demonstrate one-to-many generalization, where a single Refiner trained on small-scale open-source models successfully unlocks the latent reasoning potential of unseen models.

2 Related Work

Input Optimization for LLMs LLM performance is highly sensitive to prompt quality, driving a shift from manual engineering to automated optimization. Frameworks like TextGrad (Yuksekgonul et al., 2024) and OPRO (Yang et al., 2023) utilize gradient-like feedback to refine prompts,

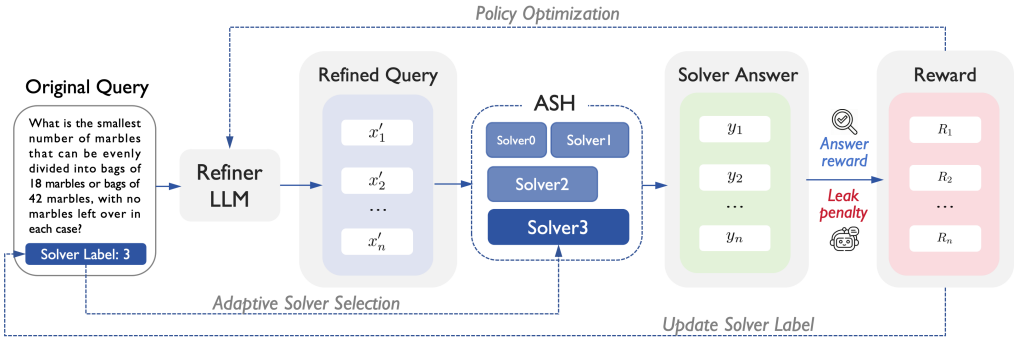

Figure 2: **ReQueR reinforcement learning pipeline.** The Refiner generates multiple refined queries for a Solver dynamically assigned by the ASH based on problem difficulty. After computing rewards from the Solver's answers, the framework optimizes the Refiner policy and updates Solver labels to ensure a stable learning curriculum.

while GEPA (Agrawal et al., 2025) employs genetic evolution. However, these methods are often task- and model-dependent, lacking the flexibility for per-sample dynamic adaptation . Alternatively, rewriting strategies such as RaR (Deng et al., 2023) and RE2 (Xu et al., 2024) rely on static, heuristic-based templates. ReQueR employs Reinforcement Learning to learn a universal meta-policy that reconstructs a problem's logical skeleton. As an adaptive inference-time frontend, it enables $O(1)$ reasoning elicitation, unlocking the latent potential of unseen Solvers and tasks.

Reinforcement Learning for Reasoning Reinforcement Learning (RL) has emerged as a dominant paradigm for enhancing the reasoning capabilities of LLMs (Gulcehre et al., 2023). Reinforcement Learning with Verifiable Rewards (RLVR) (Guo et al., 2025) leverages objective ground-truth feedback, such as mathematical correctness (Cobbe et al., 2021; Lin et al., 2026) or code execution results (Le et al., 2022), to provide precise supervision. Group Relative Policy Optimization (Shao et al., 2024; Yao et al., 2024) further improved training efficiency through group-relative advantage normalization, eliminating the need for a separate value-function critic. A broader discussion of recent RL-for-reasoning advances is provided in Appendix B. Unlike these methods that enhance Solvers directly, ReQueR optimizes a Refiner meta-policy for one-to-many reasoning elicitation, unlocking the latent potential of diverse frozen Solvers at inference time.

3 The ReQueR Framework

3.1 Problem Formulation

We reformulate the reasoning elicitation task as an **inference-time alignment** problem. Unlike SFT or RL that modifies the internal parameters of a model to match human preferences, ReQueR aligns the input distribution to match the intrinsic reasoning triggers of frozen large language models.

The Refiner and the Solver Set Let $\mathcal{D} = \{(x, y^*)\}$ be a dataset of reasoning problems where x represents the raw human inquiry and y^* is the ground-truth answer. We define the **Refiner** as a policy $\pi_\theta(x'|x)$, parameterized by θ, which transforms x into a structured, refined query x'. The environment consists of a heterogeneous set of frozen **Solvers** $\mathcal{S} = \{\mathcal{M}_k\}_{k=1}^{K}$, ranked by their intrinsic reasoning capabilities. Each Solver $\mathcal{M}_k \in \mathcal{S}$ is treated as a black box that, given an input q, generates a reasoning trajectory τ and a predicted answer $\hat{y} = \text{Extract}(\mathcal{M}_k(q))$.

The $O(1)$ Alignment Objective The core ambition of ReQueR is to learn a universal **meta-policy** that generalizes across various Solvers without requiring individual parameter updates. To formalize this "one-to-many" alignment, we define the Cross-Model Generalization performance for a given problem (x, y^*) as:

$$\mathcal{G}(\theta; x, y^*) = \mathbb{E}_{\mathcal{M}\in\mathcal{S}}[\mathcal{R}(y^*, \mathcal{M}(\pi_\theta(x)))]. \quad (1)$$

The global objective is then formulated as:

$$\max_\theta \mathbb{E}_{(x,y^*)\sim\mathcal{D}} \left[\mathcal{G}(\theta; x, y^*)\right], \quad (2)$$

where $\mathcal{R}$ is the composite reward function (Section 3.3), and the expectation over $\mathcal{S}$ is managed by the Adaptive Solver Hierarchy (Section 3.2).

Decoupling and Scalability ReQueR shifts the alignment burden from parameters to input optimization, replacing $O(N)$ model-specific tuning with a scalable $O(1)$ paradigm. By optimizing

across a Solver manifold, the Refiner internalizes universal reasoning triggers that transcend individual architectures. Once trained, π_θ can be deployed to unlock latent potential in unseen Solvers ($\mathcal{M}_{\text{unseen}}$) at inference time with zero additional training cost (Theoretical analysis in Appendix F).

Comparison with Prompt Optimization ReQueR transcends Automatic Prompt Optimization (APO) in two critical dimensions. First, while APO searches for a static, task-level wrapper string, ReQueR learns a generative meta-policy that produces a unique, tailored refinement for every instance. Second, by optimizing against a Solver manifold rather than a single model, ReQueR internalizes model-agnostic reasoning triggers. Detailed comparisons are provided in Appendix A

3.2 Environmental Dynamics via ASH

To optimize the objective defined in Equation (1), the Refiner must interact with the Solver distribution $\mathcal{S}$. However, training against a fixed Solver often leads to *reward sparsity*. A weak Solver may fail consistently regardless of query quality, while a dominant Solver might succeed even with suboptimal queries. To maintain a high-quality feedback loop, we propose the Adaptive Solver Hierarchy (ASH), a curriculum mechanism that dynamically manages environmental complexity.

Dynamic Difficulty Adjustment We rank the Solvers in $\mathcal{S}$ by capability, $\mathcal{S} = \{\mathcal{M}_1, \dots, \mathcal{M}_K\}$. Each sample $x_i \in \mathcal{D}$ is associated with a local difficulty index $k_i \in \{1, \dots, K\}$. During training, the Refiner generates a group of G queries $\{x'_{i,j}\}_{j=1}^{G}$. Let $S_i = \sum_{j=1}^{G} \mathbb{I}[\hat{y}_{i,j} = y_i^*]$ denote the success count within the group. To maintain an informative reward signal, k_i is updated for the next iteration:

$$k_i^{(t+1)} = \begin{cases} \min(K, k_i^{(t)} + 1) & \text{if } S_i = 0, \\ \max(1, k_i^{(t)} - 1) & \text{if } S_i = G, \\ k_i^{(t)} & \text{otherwise.} \end{cases} \quad (3)$$

ASH as Feedback Calibration This mechanism is conceptually grounded in the *Zone of Proximal Development* (ZPD), which posits that optimal learning occurs when the environment's difficulty is precisely calibrated to the agent's current competence. ASH operationalizes this by addressing two extremes of gradient degradation (Zhang et al., 2026). First, when $S_i = 0$, the environment is effectively "deaf" to the Refiner's structural improvements, leading to a total absence of positive reinforcement. In this scenario, ASH escalates to a more capable Solver to detect whether the refined query contains valid reasoning cues that only a stronger model can decode. Conversely, when $S_i = G$, the task is trivialized by an overly powerful Solver that bypasses the need for high-quality input. Here, ASH then de-escalates to impose "structural pressure", forcing the Refiner to generate more explicit and robust logic. By dynamically maintaining the Refiner within its ZPD, ASH prevents the policy from overfitting to a specific model's idiosyncratic tolerance and facilitates the discovery of universal reasoning triggers across the entire Solver manifold.

Algorithm 1 ReQueR RL Training Procedure

Require: $\mathcal{D}$, π_θ, π_{ref}, $\{\mathcal{M}_k\}_1^K$, $\mathcal{M}_j$
Ensure: Optimized Policy π_{θ^*}
Init: $\theta \leftarrow \theta_{\text{SFT}}$, $k_i \leftarrow \lceil K/2 \rceil \; \forall x_i \in \mathcal{D}$
for epoch $e = 1, \dots, E$ **do**
  Sample mini-batch $\mathcal{B} \subseteq \mathcal{D}$
  for each $(x_i, y_i^*) \in \mathcal{B}$ **do**
    // 1. Group Sampling & Evaluation
    $\{x'_{i,j}\}_{j=1}^{G} \sim \pi_\theta(\cdot|x_i)$
    $\hat{y}_{i,j} \leftarrow \text{Extract}(\mathcal{M}_{k_i}(x'_{i,j})), \forall j$
    $R_{i,j} \leftarrow \mathbb{I}[\hat{y}_{i,j} = y_i^*] - \lambda R_{\text{leak}}(x'_{i,j}, x_i)$
    // 2. ASH Update via Equation (3)
    Let $S_i = \sum_j \mathbb{I}[\hat{y}_{i,j} = y_i^*]$
    $\Delta k_i \leftarrow \mathbb{I}[S_i = 0] - \mathbb{I}[S_i = G]$
    $k_i \leftarrow \text{clamp}(k_i + \Delta k_i, 1, K)$
    // 3. Group Advantage Normalization
    $A_{i,j} \leftarrow (R_{i,j} - \bar{R}_i)/(\sigma(R_i) + \epsilon)$
  end for
  // 4. Gradient Update via Equation (9)
  $\theta \leftarrow \theta + \eta \nabla_\theta \mathcal{J}(\theta; \mathcal{B})$
end for

3.3 Reward Design

To ensure the Refiner learns to elicit reasoning rather than simply injecting answers, we define a composite reward $R = R_{\text{acc}} - \lambda \cdot R_{\text{leak}}$, which balances task performance with the integrity of the reasoning process.

Task Success (R_{acc}) The primary objective is the correctness of the Solver's predicted answer $\hat{y}$ compared to the ground truth y^*:

$$R_{\text{acc}} = \mathbb{I}[\hat{y} = y^*]. \quad (4)$$

Leak Penalty (R_{leak}) Information leakage occurs when the Refiner provides trivial shortcuts that bypass logical complexity. To detect such exploitation of non-reasoning shortcuts, we utilize a judge model $\mathcal{M}_j$ to measure the conditional perplexity $\text{PPL}(y^*|q)$ of the answer sequence $y^* = (t_1, \ldots, t_L)$:

$$\text{PPL}(y^*|q) = \exp\left(-\frac{1}{L}\log P_{\mathcal{M}_j}(y^*|q)\right). \quad (5)$$

We quantify leakage severity via the Perplexity Drop Ratio. To suppress reward hacking, we apply an indicator-based penalty R_{leak} that triggers when the relative shift in the answer's predictability before and after refinement exceeds a tolerance threshold τ_{leak}:

$$R_{\text{leak}} = \mathbb{I}\left[\frac{\text{PPL}(y^*|x)}{\text{PPL}(y^*|x') + \epsilon} > \tau_{\text{leak}}\right]. \quad (6)$$

This constraint acts as a semantic guardrail, compelling the Refiner to prioritize structural disambiguation over leaking surface-level cues, thereby ensuring the elicitation of genuine latent reasoning.

3.4 Training Pipeline

The training of ReQueR proceeds in two distinct stages: a supervised warm-up to establish a baseline alignment capability, followed by reinforcement learning to refine the meta-policy (Figure 2).

Phase 1: Cold-Start Bootstrapping To stabilize subsequent reinforcement learning, we initialize the Refiner π_θ via Supervised Fine-Tuning on a curated dataset $\mathcal{D}_{\text{SFT}}$ containing 6,144 high-quality (x, x') pairs. These samples are synthesized by DeepSeek-R1 (Guo et al., 2025) and further refined through rigorous manual curation to ensure logical precision. The primary objective of this phase is to align the Refiner with the required output format and establish preliminary query-refining capabilities. The policy is warmed up by minimizing the standard negative log-likelihood:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{(x,x')\sim\mathcal{D}_{\text{SFT}}}[\log \pi_\theta(x'|x)]. \quad (7)$$

Phase 2: Online Optimization Building upon the initialized policy, we employ Group Relative Policy Optimization (Guo et al., 2025) to evolve the Refiner. The advantage A_j for each refinement is computed via group normalization of the rewards:

$$A_j = \frac{R^{(j)} - \text{mean}(\{R^{(i)}\}_{i=1}^G)}{\text{std}(\{R^{(i)}\}_{i=1}^G) + \epsilon}. \quad (8)$$

The Refiner is then optimized by maximizing a clipped surrogate objective with a KL penalty. The global objective is formulated as:

$$\mathcal{J}(\theta) = \mathbb{E}_{x\sim\mathcal{D}}\Bigg[\frac{1}{G}\sum_{j=1}^{G}\min\Big(r_j A_j, \text{clip}(r_j, 1-\epsilon, 1+\epsilon)A_j\Big) - \beta\mathbb{D}_{KL}(\pi_\theta\|\pi_{\text{ref}})\Bigg], \quad (9)$$

where $r_j = \pi_\theta(x'_j|x)/\pi_{\theta_{\text{old}}}(x'_j|x)$ is the probability ratio, and the clip function restricts r_j to the interval $[1-\epsilon, 1+\epsilon]$. This process enables the Refiner to explore universal reasoning triggers. The complete procedure is summarized in Algorithm 1.

4 Experiments

4.1 Experimental Setup

Datasets and Benchmarks Training data is sourced from GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), and OpenHermes-2.5 (Teknium, 2023). We derive 6,144 high-quality samples for cold-start SFT. For the RL phase, the training set consists of 2,048 samples (1,024 each from GSM8K and MATH). To evaluate ReQueR's generalization, we employ diverse reasoning benchmarks: (i) **Mathematical Reasoning:** GSM-Symbolic (Mirzadeh et al., 2024), GSM-Plus (Li et al., 2024b), MATH-500 (Lightman et al., 2023), OlympiadBench (He et al., 2024), Omni-MATH (Gao et al., 2024), and AMC23; (ii) **General Reasoning:** MMLU-Pro (Wang et al., 2024b) and GPQA-Diamond (Rein et al., 2024).

Refiner and Solver Configurations The Refiner π_θ is based on Qwen3-4B. Training is restricted to $\mathcal{S}_{\text{train}}$ = {Qwen3-0.6B, 1.7B, 4B} (Yang et al., 2025) in non-thinking mode, while evaluation encompasses a broader $\mathcal{S}_{\text{eval}}$ to verify $O(1)$ transferability. This set features unseen dense architectures (Qwen3-8B, Llama-3.2-3B-Instruct, Llama-3.1-8B/70B-Instruct, Qwen2.5-72B-Instruct) (Yang et al., 2025) and Mixture-of-Experts paradigms (Mixtral-8x7B-Instruct (Jiang et al., 2024), DeepSeek-MoE-16b-chat (Dai et al., 2024)), testing the meta-policy's robustness across divergent model scales and structural designs.

Baseline Methods We benchmark ReQueR against five representative paradigms: (1) CoT: Standard CoT prompting; (2) Re2 (Xu et al., 2024): Heuristic input re-reading for enhanced encoding; (3) RaR (Deng et al., 2023): Human-defined query

Table 1: **Performance comparison on extended reasoning benchmarks.** ReQueR (Qwen3-4B) is trained on a fixed ASH (Qwen3-0.6B/1.7B/4B) and directly applied as a plug-and-play Refiner to all other models. This evaluates the Refiner's ability to elicit reasoning from unseen architectures. **Bold** indicates best performance; green values denote gains over zero-shot CoT.

Model	Method	AMC23	Olym. Bench	Omni-MATH	GSM-Plus	GSM-Symb.	MATH-500	GPQA-Diam.	Avg.
Llama Models									
Llama-3.2-3B-Instruct	CoT	19.69	12.76	11.04	55.43	60.64	46.60	26.77	33.28
	ReQueR	**22.19**	**19.88**	**13.48**	**65.76**	**69.38**	**50.20**	**30.30**	**38.74** (+5.47)
Llama-3.1-8B-Instruct	CoT	25.94	16.02	11.63	66.86	74.44	45.80	27.78	38.35
	ReQueR	**30.94**	**21.81**	**14.02**	**71.76**	**75.82**	**54.00**	**31.31**	**42.81** (+4.46)
Llama-3.1-70B-Instruct	CoT	50.31	31.31	17.93	80.81	85.76	64.60	47.73	54.06
	ReQueR	**53.75**	**36.05**	**20.30**	**83.38**	**87.98**	**72.00**	**48.36**	**57.40** (+3.34)
Qwen Models									
Qwen3-0.6B	CoT	18.44	19.14	13.14	44.48	50.80	48.80	27.53	31.76
	ReQueR	**23.75**	**21.66**	**15.18**	**61.57**	**66.14**	**54.40**	**30.30**	**39.00** (+7.24)
Qwen3-1.7B	CoT	41.25	37.83	20.84	69.38	74.62	64.20	31.31	48.49
	ReQueR	**43.13**	**39.02**	**22.11**	**76.71**	**78.64**	**77.60**	**32.83**	**52.86** (+4.37)
Qwen3-8B	CoT	65.31	50.00	27.91	85.67	86.96	83.40	41.92	63.02
	ReQueR	**68.13**	**51.45**	**28.16**	**88.05**	**87.08**	**85.00**	**45.45**	**64.76** (+1.74)
Qwen2.5-72B-Instruct	CoT	61.88	43.62	26.40	81.57	82.26	76.20	50.88	60.40
	ReQueR	**62.50**	**46.44**	**27.37**	**85.43**	**88.12**	**83.00**	**52.02**	**63.55** (+3.15)
Deepseek & Mixtral Models									
DS-MoE-16B-Chat	CoT	6.25	2.52	5.94	35.24	39.48	16.20	24.75	18.63
	ReQueR	**7.19**	**3.56**	**6.64**	**45.71**	**48.34**	**19.20**	**26.77**	**22.49** (+3.86)
Mixtral-8x7B-Instruct	CoT	9.69	7.12	8.63	47.29	52.12	30.60	25.76	25.89
	ReQueR	**18.12**	**10.68**	**9.28**	**57.52**	**59.66**	**33.00**	**31.31**	**31.37** (+5.48)

rephrasing and expansion; (4) TextGrad (Yuksekgonul et al., 2024): Gradient-based prompt optimization; (5) GEPA (Agrawal et al., 2025): Genetic algorithm-based prompt evolution.

Implementation and Metrics The Refiner is optimized via the verl (Sheng et al., 2025) framework (batch size 64, learning rate 1×10^{-6}). Performance is measured using the `Avg@4` accuracy at temperature 0.6. Detailed experimental configurations and hyperparameter settings are provided in Appendix C.

4.2 Main Results

Transferability across Models and Tasks Table 1 demonstrates that by training on a limited 2,048-sample subset of GSM8K and MATH using the small-scale $\mathcal{S}_{\text{train}}$, ReQueR achieves consistent performance gains across all seven reasoning benchmarks. The Refiner exhibits robust cross-model transferability, significantly elevating smaller Solvers such as Qwen3-0.6B (+7.24%) while successfully generalizing to large-scale and heterogeneous architectures, including Llama-3.1-70B-Instruct (+3.34%) and Qwen2.5-72B-Instruct (+3.15%). Notably, the Refiner also improves the MoE-based DeepSeek (+3.86%) and Mixtral (+5.48%). These results suggest that π_θ captures universal, architectural-agnostic reasoning triggers. Consequently, ReQueR establishes an efficient one-to-many inference-time paradigm for unlocking the latent potential of frozen LLMs without independent model-tuning.

Comparison with Existing Baselines ReQueR demonstrates superior task-level stability and consistent cross-model activation compared to existing paradigms. As shown in Table 2, it outperforms the strongest baseline by an average of 2.10% across all benchmarks, confirming that RL-driven refinement handles diverse logical distributions more robustly than static optimization. Furthermore, Table 3 validates its steady activation efficacy across various scales, surpassing the next best method by an average of 3.26% on MATH-500. This consistent gain from 0.6B to 72B proves that the meta-policy effectively elicits the latent potential of frozen Solvers regardless of their size or architecture.

Table 2: Performance of ReQueR on Qwen3-1.7B vs. baselines across tasks (best in **bold**, gains in green).

Method	GSM-Symb.	MATH-500	Omni-MATH	Olym. Bench	GPQA-Diam.	MMLU-Pro	Avg.
Zero-shot CoT	74.62	64.20	20.84	37.83	31.31	43.80	45.43
TextGrad	76.78	72.40	20.14	34.87	31.82	44.05	46.68
GEPA	76.95	73.60	20.62	35.12	32.05	45.32	47.28
Re2	76.78	71.40	21.03	38.13	31.31	44.01	47.11
RaR	65.64	72.80	20.37	38.58	31.82	42.98	45.37
ReQueR (Ours)	**78.64**	**77.60**	**22.11**	**39.02**	**32.83**	**46.09**	**49.38** (+2.10)

Table 3: Performance of ReQueR on MATH-500 vs. baselines across model scales (best in **bold**, gains in green).

Method	Llama-3.2-3B-Instruct	Llama-3.1-8B-Instruct	Llama-3.1-70B-Instruct	Qwen3-0.6B	Qwen3-8B	Qwen2.5-72B-Instruct	Avg.
Zero-shot CoT	46.60	45.80	64.60	48.80	83.40	76.20	60.90
TextGrad	43.00	48.20	68.00	46.80	83.00	82.20	61.87
GEPA	44.20	49.40	68.60	47.60	83.80	82.60	62.70
Re2	46.40	50.40	67.40	48.60	84.00	82.20	63.17
RaR	44.00	45.20	68.80	47.80	82.80	81.00	61.60
ReQueR (Ours)	**50.20**	**54.00**	**72.00**	**54.40**	**85.00**	**83.00**	**66.43** (+3.26)

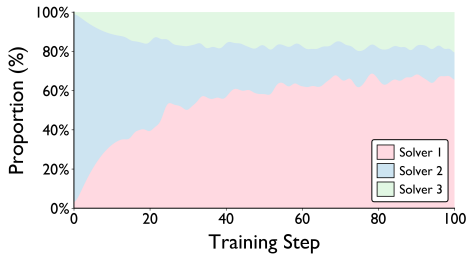

Figure 3: **Evolution of Solver distribution during training.** ASH dynamically matches Solver difficulty to the Refiner's competence.

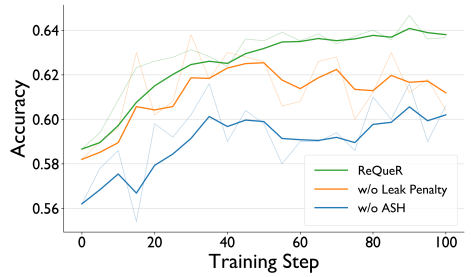

Figure 4: **Progress of validation accuracy during training.** Both ASH and Leak Penalty are necessary for maintaining stable training and robust generalization.

4.3 Further Analysis

Emergent Refinement Strategies RL training yields high-value strategies that underpin ReQueR's effectiveness. As shown in Table 5, we summarize four representative archetypes that distill human intuition into machine-executable logic: (i) Symbolic Parameterization; (ii) Contextual Denoising; (iii) Linguistic Disambiguation; and (iv) Conceptual Reduction. These emergent patterns explain ReQueR's success: they project raw queries onto a logic-dense manifold specifically tailored to trigger the Solver's latent reasoning. Full taxonomy is provided in Appendix E.

Table 4: Average response length (Tokens / Sample) of Raw queries vs. ReQueR-refined queries.

Model	Task	Raw	ReQueR	Δ (%)
Mixtral-8x7B-Instruct	GSM8K	678.0	560.8	17.3% ↓
	MATH-500	1119.2	1033.1	7.7% ↓
	AMC23	11911.5	10595.7	11.0% ↓
Qwen2.5-72B-Instruct	GSM8K	886.3	796.8	10.1% ↓
	MATH-500	1638.1	1559.0	4.8% ↓
	AMC23	19639.6	18045.5	8.1% ↓

Analysis of ASH Figure 3 reveals that ASH adaptively shifts the Solver distribution from diverse models to weaker ones as Refiner competence improves. This staged curriculum keeps the model within the Zone of Proximal Development to maintain dense rewards. As shown in Figure 4, removing ASH degrades performance significantly, confirming that balancing task difficulty with current competence is essential for stable optimization.

Analysis of Leak Penalty Figure 4 demonstrates that omitting the leakage penalty leads to an early performance surge followed by a late-stage decline. This behavior signifies reward hacking where the model exploits non-reasoning shortcuts. Our PPL-based constraint suppresses verbatim answer leakage and forces the Refiner to elicit latent logic, which ultimately yields superior generalization.

Analysis of Inference Overhead The computational overhead introduced by ReQueR depends on the deployment scenario. For static datasets,

Table 5: Case studies illustrating how ReQueR transforms raw queries into reasoning-facilitating structures.

Strategy 1: Symbolic Parameterization

Description: Transforms narrative word problems into formal symbolic models comprising variables and equations.

	Raw Query	Refined Query
Ex.	Ten treeks weigh as much as three squigs and one goolee. Two treeks and one goolee are equal in weight to one squig. The combined weight of how many treeks equals the weight of one squig?	**Let T be the weight of one treek, S be the weight of one squig, and G be the weight of one goolee.** We are given: **1.** $10T = 3S + G$**, 2.** $2T + G = S$**.** We need to **find the number** n **such that** $nT = S$. Determine n by solving the system of equations.

Strategy 2: Contextual De-noising

Description: Filters out factual but computationally irrelevant narrative noise from the problem description.

	Raw Query	Refined Query
Ex.	~~Daniel works at an electronics store, and he claims that~~ the popularity of a television is inversely proportional to its cost. If 15 customers buy a television that costs $1500, how many customers would buy a television that costs $2500?	**Given that** the number of customers buying a television is inversely proportional to its cost, and that 15 customers buy a television costing $1500, how many customers would buy a television costing $2500?

Strategy 3: Linguistic Disambiguation

Description: Resolves semantic ambiguity by replacing vague natural language with formal operators or explicit sets.

	Raw Query	Refined Query
Ex.	What is the sum of all of the multiples of 3 between 100 and 200 ?	Calculate the sum of the **arithmetic series** starting at **102**, ending at **198**, with a **common difference of 3**.

Strategy 4: Conceptual Reduction

Description: Decomposes high-level math concepts into primitive algebraic operations and decision constraints.

	Raw Query	Refined Query
Ex.	How many vertical asymptotes does the graph of $y = \frac{2}{x^2+x-6}$ have?	Find the number of **distinct real numbers** x for which the **denominator** of the function $y = \frac{2}{x^2+x-6}$ **is zero**.

refinement costs are virtually negligible as queries can be pre-processed in a single pass to serve multiple downstream Solvers. In real-time settings, although the 4B Refiner adds a secondary generation step, the overhead is limited to query reformulation, which is significantly shorter than the subsequent reasoning chains. Crucially, this latency is mitigated by improved Solver efficiency. As quantified in Table 4, refined queries yield shorter responses with length reductions between 4.8% and 17.3%. This suggests that ReQueR, by clarifying input logic, facilitates more efficient CoT trajectories that partially offset the initial refinement cost.

4.4 Ablation Study

Ablation on SFT and RL Training Table 6 reveals the cumulative gains from our progressive training pipeline. SFT establishes a foundational alignment, providing a 5.14% increase for Qwen3-0.6B on GSM-Plus, while subsequent RL further boosts performance by 11.15%. This trajectory confirms that RL is essential to fully activate the reasoning triggers initialized during SFT.

Ablation on ASH and Leak Penalty Table 6 identifies ASH as the most vital component for training stability. Its removal leads to a significant 5.85% drop for Llama-3.1-8B-Instruct on MATH-500, illustrating its role in preserving the optimal difficulty-competence balance. The leak penalty further ensures policy integrity by preventing reward hacking, as evidenced by the 2.15% decrease on GSM-Plus for Qwen3-0.6B upon its removal.

Table 6: **Comprehensive ablation study.** Impact of training stages (SFT, RL) and specific components (ASH, Leak Penalty) across different base models.

Method	MATH-500	GSM-Plus
Qwen3-0.6B	49.05	45.28
+ SFT	50.68	50.42
+ RL (Full ReQueR)	**54.40**	**61.57**
w/o ASH	52.12	54.85
w/o Leak Penalty	53.78	59.42
Llama-3.1-8B-Instruct	46.12	67.24
+ SFT	48.56	68.46
+ RL (Full ReQueR)	**54.00**	**71.76**
w/o ASH	48.15	68.32
w/o Leak Penalty	53.28	70.05
Qwen2.5-72B-Instruct	76.68	81.75
+ SFT	78.72	82.81
+ RL (Full ReQueR)	**83.00**	**85.43**
w/o ASH	78.40	82.95
w/o Leak Penalty	81.15	83.82

5 Conclusion

In this paper, we presented ReQueR, a unified inference-time Refiner policy to activate the latent reasoning capabilities of diverse, frozen LLMs. By shifting the alignment burden from individual model-tuning to a scalable $O(1)$ paradigm, our framework enables a single Refiner to elicit reasoning without Solver-parameter updates. Empiri-

cal results across diverse architectures and scales indicate the robust generalization of ReQueR, highlighting its potential as an efficient one-to-many solution for frozen LLMs. Furthermore, this approach offers significant potential for agentic workflows, such as code agents, where dynamic query refinement can effectively bridge the gap between ambiguous human intent and coding LLMs. These findings highlight ReQueR as a versatile and efficient foundation for future research into adaptive, inference-time reasoning elicitation.

Limitations

Despite its efficacy, ReQueR faces several constraints. The two-stage pipeline introduces additional latency and the risk of cascading errors, where hallucinations during refinement may mislead the Solver. Theoretically, our framework is adaptable to diverse task styles (including open-ended generation and creative writing), with the Refiner policy capable of learning specialized strategies. However, formulating rewards for non-verifiable tasks and effectively balancing reward signals across heterogeneous domains remain open challenges, currently confining our application to reasoning tasks with verifiable ground truth. Furthermore, while the ASH curriculum promotes model-agnostic triggers, the Refiner remains influenced by the architectural biases of its training pool. Finally, as an elicitor rather than a generator, ReQueR can only activate a Solver's latent potential and cannot unlock reasoning capabilities fundamentally absent from its pretrained weights.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (92570101, 62502447).

References

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, and 1 others. 2025. Gepa: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*.

Shengnan An, Xunliang Cai, Xuezhi Cao, Xiaoyu Li, Yehao Lin, Junlin Liu, Xinxuan Lv, Dan Ma, Xuanlin Wang, Ziwen Wang, and 1 others. 2025. Amo-bench: Large language models still struggle in high school math competitions. *arXiv preprint arXiv:2510.26768*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Hanyu Cai, Binqi Shen, Lier Jin, Lan Hu, and Xiaojing Fan. 2025. Does tone change the answer? evaluating prompt politeness effects on modern llms: Gpt, gemini, llama. *arXiv preprint arXiv:2512.12812*.

Yupeng Chang, Yi Chang, and Yuan Wu. 2026. BA-loRA: Bias-alleviating low-rank adaptation to mitigate catastrophic inheritance in large language models. In *The Fourteenth International Conference on Learning Representations*.

Ruijun Chen, Jiehao Liang, Shiping Gao, Fanqi Wan, and Xiaojun Quan. 2024. Self-evolution fine-tuning for policy optimization. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4120–4137.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Chongyuan Dai, Jinpeng Hu, Hongchang Shi, Zhuo Li, Xun Yang, and Meng Wang. 2025. Psyche-r1: Towards reliable psychological llms through unified empathy, expertise, and reasoning. *arXiv preprint arXiv:2508.10848*.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, and 1 others. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*.

Yihe Deng, Weitong Zhang, Zixiang Chen, and Quanquan Gu. 2023. Rephrase and respond: Let large language models ask better questions for themselves. *arXiv preprint arXiv:2311.04205*.

Ruiyi Ding, Yongxuan Lv, Xianhui Meng, Jiahe Song, Chao Wang, Chen Jiang, and Yuan Cheng. 2026. Prpo: Aligning process reward with outcome reward in policy optimization. *arXiv preprint arXiv:2601.07182*.

Haonan Dong, Kehan Jiang, Haoran Ye, Wenhao Zhu, Zhaolu Kang, and Guojie Song. 2026. Neureasoner: Towards explainable, controllable, and unified reasoning via mixture-of-neurons. *arXiv preprint arXiv:2604.02972*.

Haonan Dong, Wenhao Zhu, Guojie Song, and Liang Wang. 2025. Aurora: Breaking low-rank bottleneck of lora with nonlinear mapping. *arXiv preprint arXiv:2505.18738*.

Zhihao Dou, Qinjian Zhao, Zhongwei Wan, Dinggen Zhang, Weida Wang, Towsif Raiyan, Benteng Chen, Qingtao Pan, Yang Ouyang, Zhiqiang Gao, and 1 others. 2025. Plan then action: High-level planning guidance reinforcement learning for llm reasoning. *arXiv preprint arXiv:2510.01833*.

Yangyi Fang, Jiaye Lin, Xiaoliang Fu, Cong Qin, Haolin Shi, Chaowen Hu, Lu Pan, Ke Zeng, and Xunliang Cai. 2026a. How to allocate, how to learn? dynamic rollout allocation and advantage modulation for policy optimization. *arXiv preprint arXiv:2602.19208*.

Yangyi Fang, Jiaye Lin, Xiaoliang Fu, Cong Qin, Haolin Shi, Chang Liu, and Peilin Zhao. 2026b. Proximity-based multi-turn optimization: Practical credit assignment for llm agent training. *arXiv preprint arXiv:2602.19225*.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, and 1 others. 2024. Omni-math: A universal olympiad level mathematic benchmark for large language models. *arXiv preprint arXiv:2410.07985*.

Leo Gao, John Schulman, and Jacob Hilton. 2023. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pages 10835–10866. PMLR.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, and 1 others. 2023. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Yu Guo, Shenghao Ye, Shuangwu Chen, Zijian Wen, Tao Zhang, Qirui Bai, Dong Jin, Yunpeng Hou, Huasen He, Jian Yang, and 1 others. 2026. Rethinking table pruning in tableqa: From sequential revisions to gold trajectory-supervised parallel search. *arXiv preprint arXiv:2601.03851*.

Zhezheng Hao, Hong Wang, Haoyang Liu, Jian Luo, Jiarui Yu, Hande Dong, Qiang Lin, Can Wang, and Jiawei Chen. 2025. Rethinking entropy interventions in rlvr: An entropy change perspective. *arXiv preprint arXiv:2510.10150*.

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

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

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, and 1 others. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Kehan Jiang, Haonan Dong, Zhaolu Kang, Zhengzhou Zhu, and Guojie Song. 2026. Foe: Forest of errors makes the first solution the best in large reasoning models. *arXiv preprint arXiv:2604.02967*.

Chenghou Jin, Yixin Ren, Hongxu Ma, Yewei Xia, Yi Guan, Hao Zhang, Jiandong Ding, Jihong Guan, and Shuigeng Zhou. 2026. Invariant feature learning for counterfactual watch-time prediction in video recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 14964–14972.

Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu Hong Hoi. 2022. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. *Advances in Neural Information Processing Systems*, 35:21314–21328.

Long Li, Zhijian Zhou, Jiaran Hao, Jason Klein Liu, Yanting Miao, Wei Pang, Xiaoyu Tan, Wei Chu, Zhe Wang, Shirui Pan, and 1 others. 2025a. The choice of divergence: A neglected key to mitigating diversity collapse in reinforcement learning with verifiable reward. *arXiv preprint arXiv:2509.07430*.

Long Li, Zhijian Zhou, Tianyi Wang, Weidi Xu, Zuming Huang, Wei Chu, Zhe Wang, Shirui Pan, Chao Qu, and Yuan Qi. 2026. Dyjr: Preserving diversity in reinforcement learning with verifiable rewards via dynamic jensen-shannon replay. *arXiv preprint arXiv:2603.16157*.

Mengdi Li, Jiaye Lin, Xufeng Zhao, Wenhao Lu, Peilin Zhao, Stefan Wermter, and Di Wang. 2025b. Curriculum-rlaif: Curriculum alignment with reinforcement learning from ai feedback. *arXiv preprint arXiv:2505.20075*.

Muquan Li, Dongyang Zhang, Tao He, Xiurui Xie, Yuan-Fang Li, and Ke Qin. 2024a. Towards effective data-free knowledge distillation via diverse diffusion augmentation. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 4416–4425.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. 2024b. Gsm-plus: A comprehensive benchmark for evaluating the robustness of llms as mathematical problem solvers. *arXiv preprint arXiv:2402.19255*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Jiaye Lin, Yifu Guo, Yuzhen Han, Sen Hu, Ziyi Ni, Licheng Wang, Mingguang Chen, Hongzhang Liu, Ronghao Chen, Yangfan He, and 1 others. 2025. Seagent: Self-evolution trajectory optimization in multi-step reasoning with llm-based agents. *arXiv preprint arXiv:2508.02085*.

Zhiming Lin, Kai Zhao, Sophie Zhang, Peilai Yu, and Canran Xiao. 2026. Cec-zero: Zero-supervision character error correction with self-generated rewards. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 23612–23620.

Guoming Ling, Zhongzhan Huang, Yupei Lin, Junxin Li, Shanshan Zhong, Hefeng Wu, and Liang Lin. 2026. Neural chain-of-thought search: Searching the optimal reasoning path to enhance large language models. *arXiv preprint arXiv:2601.11340*.

Zheng Liu, Mengjie Liu, Siwei Wen, Mengzhang Cai, Bin Cui, Conghui He, and Wentao Zhang. 2025. From uniform to heterogeneous: Tailoring policy optimization to every token's nature. *arXiv preprint arXiv:2509.16591*.

Hongxu Ma, Han Zhou, Kai Tian, Xuefeng Zhang, Chunjie Chen, Han Li, Jihong Guan, and Shuigeng Zhou. 2026. Gor: A unified and extensible generative framework for ordinal regression. In *The Fourteenth International Conference on Learning Representations*.

Ziqi Miao, Haonan Jia, Lijun Li, Chen Qian, Yuan Xiong, Wenting Yan, and Jing Shao. 2026. Seeing with you: Perception-reasoning coevolution for multimodal reasoning. *arXiv preprint arXiv:2603.28618*.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2024. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*.

Jun Rao, Xuebo Liu, Hexuan Deng, Zepeng Lin, Zixiong Yu, Jiansheng Wei, Xiaojun Meng, and Min Zhang. 2025. Dynamic sampling that adapts: Iterative dpo for self-aware mathematical reasoning. *arXiv preprint arXiv:2505.16176*.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Tao Ren, Jinyang Jiang, Hui Yang, Wan Tian, Minhao Zou, Guanghao Li, Zishi Zhang, Qinghao Wang, Shentao Qin, Yanjun Zhao, and 1 others. 2025. Riskpo: Risk-based policy optimization via verifiable reward for llm post-training. *arXiv preprint arXiv:2510.00911*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pages 1279–1297.

Teknium. 2023. Openhermes 2.5: An open dataset of synthetic data for generalist llm assistants.

Lev S Vygotsky. 1978. *Mind in society: The development of higher psychological processes*, volume 86. Harvard university press.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and 1 others. 2024a. Large language models are not fair evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9440–9450.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, and 1 others. 2024b. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Fei Wu, Zhenrong Zhang, Qikai Chang, Jianshu Zhang, Quan Liu, and Jun Du. 2026. Step potential advantage estimation: Harnessing intermediate confidence and correctness for efficient mathematical reasoning. *arXiv preprint arXiv:2601.03823*.

Zhenhe Wu, Jian Yang, Jiaheng Liu, Xianjie Wu, Changzai Pan, Jie Zhang, Yu Zhao, Shuangyong Song, Yongxiang Li, and Zhoujun Li. 2025. Table-r1: Region-based reinforcement learning for table understanding. *arXiv preprint arXiv:2505.12415*.

Canran Xiao, Jiabao Dou, Zhiming Lin, Zong Ke, and Liwei Hou. 2026a. From points to coalitions: hierarchical contrastive shapley values for prioritizing data

samples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 15995–16003.

ShiLin Xiao, Tianxiang Xu, Canran Xiao, Weihao Luo, Liwei Hou, and Chuangxin Zhao. 2026b. Meta-ucf: Unified task-conditioned lora generation for continual learning in large language models. In *The Fourteenth International Conference on Learning Representations*.

Xi Xiao, Yunbei Zhang, Xingjian Li, Tianyang Wang, Xiao Wang, Yuxiang Wei, Jihun Hamm, and Min Xu. 2025. Visual instance-aware prompt tuning. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 2880–2889.

Can Xie, Ruotong Pan, Xiangyu Wu, Yunfei Zhang, Jiayi Fu, Tingting Gao, and Guorui Zhou. 2025. Unlocking exploration in rlvr: Uncertainty-aware advantage shaping for deeper reasoning. *arXiv preprint arXiv:2510.10649*.

Xiaohan Xu, Chongyang Tao, Tao Shen, Can Xu, Hongbo Xu, Guodong Long, Jian-Guang Lou, and Shuai Ma. 2024. Re-reading improves reasoning in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15549–15575.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2023. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*.

Jiawei Yao, Chuming Li, and Canran Xiao. 2024. Swift sampler: Efficient learning of sampler by 10 parameters. *Advances in Neural Information Processing Systems*, 37:59030–59053.

Shenghao Ye, Yu Guo, Dong Jin, Yikai Shen, Yunpeng Hou, Shuangwu Chen, Jian Yang, and Xiaofeng Jiang. 2025. When tableqa meets noise: A dual denoising framework for complex questions and large-scale tables. *arXiv preprint arXiv:2509.17680*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Zhiqi Yu, Zhangquan Chen, Mengting Liu, Heye Zhang, and Liangqiong Qu. 2026. Unveiling implicit advantage symmetry: Why grpo struggles with exploration and difficulty adaptation. *arXiv preprint arXiv:2602.05548*.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. 2024. Textgrad: Automatic" differentiation" via text. *arXiv preprint arXiv:2406.07496*.

Xinglang Zhang, Yunyao Zhang, ZeLiang Chen, Junqing Yu, Wei Yang, and Zikai Song. 2026. Logical phase transitions: Understanding collapse in llm logical reasoning. *arXiv preprint arXiv:2601.02902*.

Kai Zhao, Yanjun Zhao, Jiaming Song, Shien He, Lusheng Zhang, Qiang Zhang, and Tianjiao Li. 2026a. Saber: Switchable and balanced training for efficient llm reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 34950–34958.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International conference on machine learning*, pages 12697–12706. PMLR.

Ziqi Zhao, Zhaochun Ren, Jiahong Zou, Liu Yang, Zhiwei Xu, Xuri Ge, Zhumin Chen, Xinyu Ma, Daiting Shi, Shuaiqiang Wang, and 1 others. 2026b. Reinforced efficient reasoning via semantically diverse exploration. *arXiv preprint arXiv:2601.05053*.

Yixiao Zhou, Yang Li, Dongzhou Cheng, Hehe Fan, and Yu Cheng. 2026. Look inward to explore outward: Learning temperature policy from llm internal states via hierarchical rl. *arXiv preprint arXiv:2602.13035*.

Chunzheng Zhu, Yangfang Lin, Shen Chen, Yijun Wang, and Jianxin Lin. 2026. Medeyes: Learning dynamic visual focus for medical progressive diagnosis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 13916–13924.

Chunzheng Zhu, Yangfang Lin, Jialin Shao, Jianxin Lin, and Yijun Wang. 2025. Pathology-aware prototype evolution via llm-driven semantic disambiguation for multicenter diabetic retinopathy diagnosis. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 9196–9205.

A Paradigm Comparison: ReQueR vs. Prompt Optimization

To further clarify the novelty of ReQueR, we formalize the paradigmatic divergence between our proposed instance-level refinement and Automatic Prompt Optimization (APO).

A.1 APO as a Static Task-level Wrapper

APO methods (e.g., OPRO, Textgrad, GEPA) treat the prompt P as a discrete, static hyperparameter tailored for a specific model M_i on a task distribution $\mathcal{D}_k$. This approach can be formulated as a

point-wise search for a global string:

$$\hat{P}_{\text{APO}} = \arg\max_{P} \mathbb{E}_{x\sim\mathcal{D}_k}\Big[\mathcal{R}\left(M_i(P \oplus x), y^*\right)\Big], \tag{10}$$

where $\oplus$ denotes string concatenation. Here, P acts as a fixed wrapper, a "one-size-fits-all" instruction prefix. Because this wrapper is tightly coupled to model M_i's internal biases, it lacks the flexibility to adapt to per-sample nuances, leading to an $O(N)$ migration complexity where N is the number of target Solvers.

A.2 ReQueR as an Instance-level Meta-Policy

In contrast, ReQueR optimizes a generative policy π_θ that produces a unique refinement x' for every unique input x. By optimizing against the Solver manifold $\mathcal{S}$ via the ASH hierarchy, the Refiner learns a universal mapping:

$$\theta^* = \arg\max_{\theta} \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{s\sim\mathcal{S}}\mathbb{E}_{x'\sim\pi_\theta(x'|x)}\Big[\mathcal{R}\left(s(x'), y^*\right)\Big]. \tag{11}$$

Unlike APO, the optimization target here is a **Meta-refinement Policy**. This shift yields several critical advantages:

- **Per-sample Elasticity:** For every input x, the Refiner generates a tailored x' that rectifies the specific logical vulnerabilities of that sample, acting as a dynamic transformation rather than a static bandage. Related per-sample strategies have been explored in visual prompt tuning (Xiao et al., 2025) and SFT data construction (Chen et al., 2024).
- **Model-Agnostic Invariance:** Since the policy is trained against the distribution $\mathcal{S}$, it internalizes a meta-strategy for query clarification that is effective across all LLM architectures, enabling $O(1)$ universal transfer.
- **Inference-time Practicality and Robustness:** A significant limitation of APO is its dependence on a priori knowledge of the model and task category during inference. In real-world scenarios, one cannot assume the specific Solver identity or that a query belongs to a singular, predefined task distribution. APO's static wrappers often fail when encountering hybrid-style queries or out-of-distribution (OOD) inputs. In contrast, ReQueR's meta-policy treats every input as a fresh structural disambiguation task, ensuring high-fidelity reasoning elicitation regardless of the underlying task category or Solver backend.

The core conceptual differences are summarized in Table 7.

B Extended Related Work on RL for Reasoning

Beyond the core RL-for-reasoning line discussed in the main paper, recent advances have explored several orthogonal directions. On reward and advantage design, a range of studies investigate dynamic sampling, credit assignment, step-level advantage estimation, and risk-aware objectives to produce denser and more informative training signals (Rao et al., 2025; Fang et al., 2026b,a; Wu et al., 2026; Ren et al., 2025). For exploration and efficiency, works target token-level policy tailoring, uncertainty-aware advantage shaping, diversity-promoting exploration, inference-time temperature control, balanced training pipelines, and planning-guided RL (Liu et al., 2025; Xie et al., 2025; Zhao et al., 2026b; Zhou et al., 2026; Zhao et al., 2026a; Dou et al., 2025). Another line extends RL-for-reasoning beyond pure text, covering multimodal perception-reasoning coevolution and neural chain-of-thought search (Miao et al., 2026; Ling et al., 2026). Finally, domain-specific applications demonstrate the broader impact of RL-based reasoning in psychological, tabular, and medical settings (Dai et al., 2025; Guo et al., 2026; Ye et al., 2025; Wu et al., 2025; Zhu et al., 2026, 2025). These works collectively enrich the RL-for-reasoning landscape and motivate ReQueR's orthogonal focus on inference-time, model-agnostic reasoning elicitation.

C Experimental Details and Hyperparameters

C.1 Model Specifications and Sources

To ensure transparency and reproducibility, we provide the detailed configurations and official sources of the models utilized in the ReQueR framework. Table 8 summarizes the Refiner and the heterogeneous Solver pool, highlighting the diversity in architecture and scale that underpins our one-to-many generalization claims.

Table 7: Conceptual Comparison: APO vs. ReQueR.

Dimension	Automatic Prompt Optimization	ReQueR (Meta-Policy)
Optimization Goal	Best wrapper string for a whole task	**Meta-refinement policy** for every input x
Granularity	Global / Dataset-level	Local / Instance-level
Mechanism	Static Prefixing ($P \oplus x$)	Dynamic Transformation ($f : x \to x'$)

Table 8: Summary of Model Architectures and Official Sources.

Role	Model	Official Source / Hugging Face Link
Refiner	Qwen3-4B	Qwen/Qwen3-4B
Solvers	Qwen3-0.6B*	Qwen/Qwen3-0.6B
	Qwen3-1.7B*	Qwen/Qwen3-1.7B
	Qwen3-4B*	Qwen/Qwen3-4B
	Qwen3-8B*	Qwen/Qwen3-8B
	Qwen2.5-72B-Instruct	Qwen/Qwen2.5-72B-Instruct
	Llama-3.2-3B-Instruct	meta-llama/Llama-3.2-3B-Instruct
	Llama-3.1-8B-Instruct	meta-llama/Llama-3.1-8B-Instruct
	Llama-3.1-70B-Instruct	meta-llama/Llama-3.1-70B-Instruct
	DeepSeek-MoE-16B-Chat	deepseek-ai/deepseek-moe-16b-chat
	Mixtral-8x7B-Instruct	mistralai/Mixtral-8x7B-v0.1

* Non-thinking mode (To evaluate the direct reasoning elicitation effect without internal CoT buffers.).

C.2 Dataset Specifications and Benchmarks

To evaluate the reasoning elicitation and generalization capabilities of ReQueR, we utilize a comprehensive set of datasets for both policy training and multi-domain evaluation. Table 9 summarizes the scope, specific tasks, and official sources for each dataset used in our experiments.

C.3 Training Infrastructure

All experiments are conducted on a cluster of 8 NVIDIA H100 GPUs. To balance the computational load between policy evolution and environment feedback:

- **Actor/Refiner**: Hosted on GPUs 0–3 for active policy gradient updates.
- **Evaluator/Solvers/Leak Judge**: Hosted on GPUs 4–7 to perform parallel inference for reward computation.

The training pipeline is implemented using PyTorch and the verl framework. We employ the GRPO algorithm for policy updates and utilize vLLM for accelerated offline reward inference.

C.4 Hyperparameter Settings

The detailed hyperparameters for the RL stage, ASH mechanism, and Leak Penalty are summarized in Table 10.

C.5 Prompt Templates

To ensure the reproducibility of our reasoning elicitation process, we detail the specific prompts used for both the Refiner and the downstream Solvers. Table 11 illustrates the Refiner's meta-policy, while Table 12 defines the standardized evaluation environment for the Solvers.

D Additional Experimental Analysis

To further characterize the behavior of ReQueR, we provide a set of supplementary analyses covering weak-to-strong transfer, cross-family curriculum, the role of supervised warm-up, the robustness of the leakage penalty, and failure-mode diagnostics.

D.1 Weak-to-Strong Transfer Matrix

While the main results (Table 1) already implicitly demonstrate weak-to-strong transfer—the training pool caps at Qwen3-4B yet the Refiner consistently improves Solvers up to Qwen2.5-72B—we provide a controlled single-Solver analysis here. We train three variants of the Refiner, each using exactly one Solver during RL (Qwen3-0.6B, Qwen3-1.7B, or Qwen3-4B), and evaluate every variant on three unseen stronger Solvers across three benchmarks.

Several observations emerge. First, even the Only-0.6B Refiner—trained exclusively against a 0.6B Solver—improves Qwen2.5-72B by roughly $+3.9\%$ on average, spanning a $120\times$ scale gap.

Table 9: Summary of Datasets for Training and Evaluation.

Phase	Dataset	Samples / Scope	Official Source / Hugging Face Link
Training	OpenHermes-2.5 (Teknium, 2023)	6,144 (SFT)	teknium/OpenHermes-2.5
	GSM8K (Cobbe et al., 2021)	1,024 (RL)	openai/gsm8k
	MATH (Hendrycks et al., 2021)	1,024 (RL)	hendrycks/competition_math
Evaluation (Mathematical)	MATH-500 (Lightman et al., 2023)	500 (Hard Math)	HuggingFaceH4/MATH-500
	GSM-Symbolic (Mirzadeh et al., 2024)	5,000	apple/GSM-Symbolic
	GSM-Plus (Li et al., 2024b)	2,400	qintongli/GSM-Plus
	OlympiadBench (He et al., 2024)	674	Hothan/OlympiadBench
	Omni-MATH (Gao et al., 2024)	4,428	KbsdJames/Omni-MATH
	AMC2023	Competition Math	math-ai/amc23
Evaluation (General)	MMLU-Pro (Wang et al., 2024b)	12,000	TIGER-Lab/MMLU-Pro
	GPQA Diamond (Rein et al., 2024)	198	Idavidrein/gpqa

Table 10: Hyperparameters for ReQueR Training.

Category	Hyperparameter	Value
RL (GRPO)	Learning Rate	1×10^{-6}
	Global Batch Size	64
	PPO Mini-batch Size	32
	Micro-batch Size per GPU	8
	KL Coefficient (β)	0.05
	Entropy Coefficient	0
	Rollout Variants (G)	8
	Total Training Epochs	6
ASH Mechanism	Solver1	Qwen3-0.6B
	Solver2	Qwen3-1.7B
	Solver3	Qwen3-4B
	Initial Solver Label	$n_{\text{Solver}}//2$
Leak Penalty	Perplexity Threshold (τ_{leak})	5.0
	Penalty Coefficient (λ)	1.0
	Score Deduction	-1.0

This indicates that the refinement policy captures universal reasoning triggers rather than model-specific shortcuts. Second, every single-Solver variant outperforms the CoT baseline on all unseen Solvers, confirming that the refinement generalizes upward regardless of training-Solver strength. Third, the multi-Solver ASH curriculum consistently achieves the best overall accuracy, validating its role in discovering more robust, architecture-agnostic triggers.

D.2 Cross-Family Adaptive Solver Hierarchy

Our main ASH uses Qwen3-0.6B/1.7B/4B to obtain a strictly monotonic capability ordering, which simplifies the curriculum update rule in Equation 3. To verify that this design does not implicitly encode family-specific biases, we train a cross-family variant (ASH-CF) that replaces Qwen3-0.6B with Llama-3.2-3B-Instruct and keeps Qwen3-1.7B and Qwen3-4B unchanged.

The two configurations perform comparably (64.04 vs. 64.31 on average), with ASH-CF offering marginal gains on cross-family Solvers such as Llama-3.1-70B and Qwen2.5-72B. This suggests that single-family ASH already captures architecture-agnostic reasoning triggers, and cross-family diversity primarily yields incremental benefits rather than correcting a systematic bias. We therefore keep the single-family design in the main experiments for training simplicity.

D.3 Role of SFT: From-Scratch RL and Data Composition

The cold-start SFT dataset is composed as follows: 4,096 samples from OpenHermes-2.5, 1,024 samples from GSM8K (train), and 1,024 samples from MATH (train), yielding 6,144 samples in total. OpenHermes provides general-domain refinement patterns to prevent the Refiner from overfitting to mathematical phrasing, while the math subsets align the Refiner with the downstream RL training distribution.

Importantly, SFT in ReQueR is not the main source of reasoning capability; it mainly serves as format alignment, teaching the `<think>...</think><rephrase>...</rephrase>` output structure. To verify this, we train a Refiner using RL only, without any SFT initialization, and evaluate it across six benchmarks (MATH-500, GSM-Plus, GSM-Symbolic, OlympiadBench, AMC23, GPQA-Diamond) and three unseen stronger Solvers (Qwen3-8B, Llama-3.1-70B, Qwen2.5-72B).

From-scratch RL already closes most of the gap to full ReQueR, indicating that the reasoning-eliciting behavior is primarily acquired during the RL stage, and that ReQueR does not fundamentally depend on high-quality distillation data such as that generated by DeepSeek-R1.

Table 11: Refiner Prompt Template (SFT and RL Phases).

System Prompt

 # Role

You are an expert **Reasoning Query Refiner**. Your core task is **not** to answer questions directly, but to act as a "cognitive frontend" between a human user and a downstream AI Solver. You must rewrite the user's Raw Query into a Refined Query that is semantically equivalent but formulated in a way that maximizes the Solver's success rate.

 # Objective

Translate natural language ambiguity into machine-solvable logic. Disambiguate intent, make implicit constraints explicit, and structure the query to trigger the Solver's latent reasoning capabilities.

 # Input/Output Format

Input: User's raw natural language query.

Output: First, analyze how to refine the problem step by step in `<think>` tags, then provide the refined query in `<rephrase>` tags.

Format:

```
<think>(step by step thinking on how to rephrase questions)</think><rephrase>(rephrased version of the problem)</rephrase>
```

User Prompt Template

Original Problem:

```
{original_question}
```

Rephrase Problem:

Table 12: Solver Prompt Template and Standardized Input Format.

System Prompt

You are a math assistant. Please solve the following problem step by step and put your final answer in \boxed{} format.

User Input

Question:

```
{problem}
```

Answer:

Table 13: Weak-to-strong transfer across single-Solver Refiners and ASH. Every variant, including one trained solely on a 0.6B Solver, improves Solvers up to $120\times$ larger.

Task	Test Solver	CoT	Only-0.6B	Only-1.7B	Only-4B	ASH
MATH-500	Qwen3-8B	83.40	83.85	83.65	83.25	**85.00**
	Llama-3.1-70B	64.60	71.25	70.75	70.85	**72.00**
	Qwen2.5-72B	76.20	79.85	78.40	80.10	**83.00**
GSM-Symb.	Qwen3-8B	86.96	87.18	87.38	87.64	**87.08**
	Llama-3.1-70B	85.76	86.54	86.70	86.96	**87.98**
	Qwen2.5-72B	82.26	87.72	87.98	87.38	**88.12**
Olym.Bench	Qwen3-8B	50.00	51.48	51.11	51.59	**51.45**
	Llama-3.1-70B	31.31	35.76	35.50	35.13	**36.05**
	Qwen2.5-72B	43.62	46.28	45.48	46.11	**46.44**
Avg.		67.12	69.99	69.66	69.89	**70.79**

D.4 Leakage Penalty: Threshold Selection and Robustness

Principled threshold selection. We construct a controlled leakage-detection benchmark by pairing positive samples (legitimate refinements) with negative samples (refinements with injected answer leakage) on MATH, and evaluate precision, recall, and F1 across candidate thresholds.

We further validate this choice by training separate Refiners with $\tau \in \{2.0, 3.0, 5.0, 7.0\}$ and evaluating across five benchmarks and five Solvers.

The results remain stable in the $\tau \in [2.0, 7.0]$ range, confirming robustness to the specific threshold and supporting the use of $\tau = 5.0$ in the main

Table 14: Cross-family ASH (ASH-CF) compared with the single-family ASH across three benchmarks and six Solvers.

Task	Method	Qwen3-0.6B	Qwen3-1.7B	Qwen3-8B	Llama-3.2-3B-Inst.	Llama-3.1-70B-Inst.	Qwen2.5-72B-Inst.	Avg.
GSM-Symb.	ASH	66.14	78.64	87.08	69.38	87.98	88.12	79.56
	ASH-CF	66.06	79.10	87.74	69.84	87.86	88.38	79.83
GSM-Plus	ASH	61.57	76.71	88.05	65.76	83.38	85.43	76.82
	ASH-CF	64.00	77.38	86.52	64.81	84.62	85.86	77.20
Olym.Bench	ASH	21.66	39.02	51.45	19.88	36.05	46.44	35.75
	ASH-CF	20.88	38.98	51.34	20.36	36.50	47.26	35.89
Avg. (ASH)		49.79	64.79	75.53	51.67	69.14	73.33	64.04
Avg. (ASH-CF)		50.31	65.15	75.20	51.67	69.66	73.83	64.31

Table 15: Effect of the SFT warm-up stage. From-scratch RL alone already achieves substantial gains; SFT accelerates convergence but is not essential.

Method	Avg. Accuracy
CoT (baseline)	65.01
From-Scratch RL (no SFT)	67.72 (+2.71)
Full ReQueR (SFT + RL)	**68.01** (+3.00)

experiments.

Detecting paraphrased leakage. A key advantage of the perplexity-drop ratio over rule-based string matching is the ability to detect semantically paraphrased leakage. Consider the problem "Let $p(x) = x^3 - 3x + 1$. How many real roots does $p(p(x)) = 0$ have?", whose ground-truth answer is 7. For three paraphrased leakage variants, the ratio is consistently well above $\tau = 5.0$:

Naturally predictable answers. For well-posed queries with inherently predictable answers, the penalty does not produce false positives. For example, with "What is $1 + 1$?" rephrased as "Compute the sum of 1 and 1.", we obtain $\text{PPL}(y^*|x) = 1.26$ and $\text{PPL}(y^*|x') = 1.08$, giving a ratio of $1.17 \ll \tau = 5.0$. Because the penalty measures the *relative* change in answer predictability rather than the absolute level, already-unambiguous queries cannot be incorrectly flagged as leakage.

D.5 Failure Mode Diagnosis

To understand when ReQueR fails, we manually categorize all failure cases (reward $= 0$) on the MATH-500 validation set.

The majority of failures (85.2%) are not caused by refinement errors: the refined query is structurally sound, but the problem's difficulty exceeds the Solver's latent reasoning ability. This directly aligns with our Limitations section, which states that ReQueR can only activate latent potential rather than create reasoning capabilities absent from the Solver. The remaining failures are bounded and predictable: diagram-dependent problems whose reasoning hinges on visual elements, and a small fraction of low-quality outputs.

D.6 Self-Improving Refiner via Emergent Strategies

As an exploratory extension, we test whether the emergent refinement strategies discovered by the Refiner itself (Tables 5 and 21) can be fed back into its own prompt to form a self-improving loop. We inject the seven emergent strategies as explicit heuristic guidance into the Refiner's system prompt and retrain; evaluation uses Qwen3-0.6B as the Solver.

The Strategy Prompt improves performance across all five benchmarks, suggesting that the Refiner's discoveries can be recycled to bootstrap an even stronger Refiner. We view this as a promising direction for future work on recursive self-improvement of inference-time meta-policies.

E Qualitative Analysis: Refinement Archetypes and Case Studies

E.1 Comprehensive Taxonomy of Refinement Strategies

To investigate the operational mechanisms of the Refiner's meta-policy, we perform a qualitative decomposition of the generated queries across diverse reasoning tasks. Table 21 provides a comprehensive taxonomy of seven emergent strategies that facilitate the transition from raw human intuition to machine-executable logic. These strategies, ranging from *Symbolic Parameterization* to *Chain-of-Thought Structuring*, do not merely rephrase the

Table 16: Detection performance of the perplexity-drop ratio across thresholds. $\tau = 5.0$ achieves the peak F1.

Metric	1.5	2.0	3.0	4.0	5.0	7.0	10.0
Precision	0.828	0.885	0.947	0.967	**0.978**	0.979	0.981
Recall	0.963	0.927	0.891	0.873	0.868	0.697	0.502
F1	0.890	0.906	0.918	0.918	**0.920**	0.814	0.664

Table 17: Downstream sensitivity to the leakage threshold τ. Performance is stable across $\tau \in [2.0, 7.0]$, with $\tau = 5.0$ giving the best average.

τ	**Qwen3-1.7B**	**Qwen3-4B**	**Qwen3-8B**	**Llama-3.1-70B**	**Qwen2.5-72B**	**Avg.**
2.0	60.65	68.91	70.61	64.99	70.86	67.20
3.0	60.03	68.82	69.32	64.51	70.82	66.70
5.0	60.96	68.83	**71.41**	**65.55**	**71.00**	**67.55**
7.0	61.20	68.35	70.15	65.54	69.77	67.00

Table 18: Paraphrased leakage is reliably detected through perplexity-drop, regardless of surface form.

Paraphrased Leakage	PPL$(y^*\|x)$	PPL$(y^*\|x')$	**Ratio**	**Detected**
"The answer equals $2^3 - 1$."	35.81	2.05	17.5	✓
"It can be shown that the answer is seven."	35.81	1.10	32.6	✓
"Since p has three real roots...giving $3 + 3 + 1 = 7$ total."	35.81	1.83	19.6	✓

Table 19: Failure-mode breakdown on MATH-500. The dominant mode reflects inherent Solver capability limits rather than Refiner errors.

Failure Category	**Proportion**	**Description**
Valid rephrase, Solver still fails	85.2%	Refinement is structurally sound; problem exceeds the Solver's capability.
Diagram-dependent problems (`[asy]`)	10.9%	Visual information encoded in Asymptote blocks challenges the text-only Refiner.
Low-quality rephrase	3.9%	Overly brief or verbose output, near-verbatim repetition, or formatting issues.

Table 20: Recursive refinement: using the Refiner's own emergent strategies as prompt guidance yields further gains.

Benchmark	**Simple Prompt**	**Strategy Prompt**	Δ
AMC23	23.75	25.62	+1.87
Olym.Bench	21.66	24.81	+3.15
GSM-Plus	61.57	64.76	+3.19
GSM-Symb.	66.14	66.54	+0.40
MATH-500	54.40	55.85	+1.45

input text but perform a *Structural Transduction* of the problem space. By rectifying specific logical vulnerabilities and making implicit constraints explicit, these archetypes ensure that the refined query is projected onto a logic-dense manifold, effectively eliciting the latent reasoning capabilities of downstream Solvers regardless of their architecture or scale.

E.2 Generalization to Unseen Domains

The effectiveness of the Refiner meta-policy is highly suggestive of its capacity to internalize a domain-invariant reasoning syntax. Rather than merely memorizing task-specific patterns, the Refiner appears to utilize mathematical datasets such as GSM8K and MATH as a rigorous training ground to master structural primitives that are foundational to logical discourse. This cross-domain adaptability is empirically illustrated in Table 22, which details representative case studies curated from the MMLU-Pro benchmark. These examples demonstrate the Refiner's ability to perform Structural Transduction across diverse fields including law, biology, and computer science.

By decoupling the process of reasoning from the repository of domain knowledge, the Refiner functions as a versatile cognitive frontend that prioritizes formal clarity over semantic surface. As observed in the MMLU-Pro evaluations, the symbolic

parameterization strategy originally developed for mathematical word problems is functionally parallel to the process of isolating editorial discretion in a legal dispute, as both operations require the Refiner to strip away semantic noise to reveal the underlying logical constraints. Consequently, ReQueR establishes a standardized interface that enables downstream Solvers to mitigate linguistic ambiguity and more effectively activate their latent logical capabilities across the entire MMLU-Pro spectrum. This zero-shot generalization indicates that the Refiner has moved beyond simple pattern matching toward a deeper, model-agnostic understanding of problem decomposition.

F Theoretical Framework

The ReQueR framework establishes a formal paradigm for reasoning elicitation by decoupling the alignment process from the Solver's internal parametric space. We justify this approach through the dual lenses of Amortized Complexity and Information-Theoretic Feedback Calibration.

F.1 The O(1) Paradigm: Amortized Alignment via Input-Side Mapping

Traditional alignment methodologies, such as Supervised Fine-Tuning or Reinforcement Learning, operate primarily within the Parametric Manifold Θ. For a heterogeneous ensemble of N Solvers $\mathcal{S} = \{s_1, \ldots, s_N\}$, where each s_i is parameterized by $\theta_i \in \Theta_i$, the total parametric alignment cost $\mathcal{C}_{\text{param}}$ scales linearly with N:

$$\mathcal{C}_{\text{param}} = \sum_{i=1}^{N} \int_{t=0}^{T} \|\nabla_{\theta_i} \mathcal{J}(\theta_i, \mathcal{D})\| dt \in O(N), \tag{12}$$

where $\mathcal{J}$ is the alignment objective and T is the optimization horizon.

Existing Automated Prompt Optimization techniques mitigate this overhead by performing a Point-Estimate Optimization over the discrete token space $\mathcal{V}^*$. APO seeks a static wrapper string p^* for a specific task distribution $\mathcal{D}_k$:

$$p^*_{APO} = \arg\max_{p \in \mathcal{V}^*} \mathbb{E}_{x \sim \mathcal{D}_k}[\mathcal{R}(s(p \oplus x))], \tag{13}$$

where $\oplus$ denotes string concatenation. While parameter-efficient, this "one-size-fits-all" wrapper lacks per-sample granularity and is often tightly coupled to the internal biases of a specific Solver, limiting its zero-shot migration capability.

In contrast, ReQueR reformulates alignment as an Amortized Mapping problem $\Phi : \mathcal{X} \to \mathcal{X}'$, where $\mathcal{X}'$ is a reasoning-dense manifold. Let $\pi_\theta(x'|x)$ be the Refiner policy. We define the universal alignment objective as maximizing the expectation over both the task distribution $\mathcal{D}$ and the Solver manifold $\mathcal{S}$:

$$\max_\theta \mathcal{J}(\theta) = \mathbb{E}_{x \sim \mathcal{D}} \Bigg[\int_{s \in \mathcal{S}} \mathbb{E}_{x' \sim \pi_\theta(x'|x)} \Big[\mathcal{R}(s(x')) \Big] \, d\mu(s) \Bigg], \tag{14}$$

where $\mu(s)$ is a measure over the Solver space. Since θ is optimized once and remains invariant across all $s \in \mathcal{S}$, the complexity relative to the number of Solvers becomes $\mathcal{C}_{\text{ReQueR}} \in O(1)$. The Refiner functions as an Invariant Canonical Projection (Jin et al., 2026; Ma et al., 2026), extracting structural logical primitives that remain cross-architecturally effective.

F.2 ASH: Optimization Stability via Gradient Signal Recovery

The Adaptive Solver Hierarchy serves as a strategic framework for Gradient Signal Recovery. Under the GRPO framework, the policy gradient $\nabla_\theta \mathcal{J}(\theta)$ is driven by the group-relative advantage A_i:

$$\nabla_\theta \mathcal{J}(\theta) \approx \frac{1}{G} \sum_{i=1}^{G} \nabla_\theta \log \pi_\theta(x'_i|x) \cdot \frac{R_i - \bar{R}}{\sigma_R + \epsilon}, \tag{15}$$

where $\bar{R} = \frac{1}{G}\sum R_i$ and σ_R is the group standard deviation.

Reward Entropy Collapse and SNR For a binary reward $\mathcal{R} \in \{0, 1\}$, the informativeness of the update depends on the Signal-to-Noise Ratio (SNR) of the advantage signal:

$$\text{SNR} = \frac{\text{Var}(\mathbb{E}[R|\pi_\theta])}{\mathbb{E}[\text{Var}(R|\pi_\theta)] + \sigma^2_{\text{noise}}}, \tag{16}$$

When the Solver s_k is poorly matched to the task difficulty (e.g., an "Expert" where $P \to 1$ or a "Novice" where $P \to 0$), the reward distribution suffers from Entropy Collapse (Hao et al., 2025), i.e., $\mathbb{H}(\mathcal{R}) \to 0$. Consequently, $\sigma_R \to 0$, leading to a vanishing SNR and gradient stagnation.

Optimal Calibration via ZPD Constraints ASH operationalizes the Zone of Proximal Development by dynamically selecting a Solver s_{k^*} that

acts as a Maximum Information Elicitor. We define the optimal Solver index at time t as:

$$k^*(t) = \arg\min_k \Big\| \mathbb{E}_{x\sim\mathcal{D}} [P(y = y^*|x, s_k, \pi_\theta)] - \tau \Big\|_2, \tag{17}$$

where $\tau \in (0, 1)$ is the target difficulty threshold (typically $\tau \approx 0.5$ to maximize the variance $\tau(1 - \tau)$). By maintaining the Refiner at its ZPD edge, ASH ensures Continuous Gradient Pressure, forcing the synthesis of "Hardened Triggers" that possess high logical density and are cross-architecturally effective.

G Promising Application Scenarios

Beyond the benchmarks presented in this study, the $O(1)$ alignment paradigm of ReQueR offers significant potential as a Cognitive Frontend that bridges the gap between raw human intent and the specialized execution logic of heterogeneous Solvers. By decoupling *reasoning elicitation* (Dong et al., 2026) from *knowledge retrieval*, ReQueR enables a scalable Edge-Cloud Collaborative Inference architecture where a centralized, high-capacity Refiner empowers a diverse fleet of edge-deployed models without individual fine-tuning.

Autonomous Code Agents as Requirement Architects In the domain of complex software engineering, ReQueR transcends simple query rephrasing by functioning as a *Requirement Architect*. By identifying implicit logical constraints and structural dependencies within ambiguous user specifications, the Refiner projects raw natural language into a "logic-dense" manifold. This process significantly minimizes the search space for downstream code-generation Solvers and mitigates the hallucination of non-existent API parameters (Jiang et al., 2026), allowing even small-scale code models to execute high-level architectural designs with precision.

Agentic Workflow Orchestration For multi-agent systems requiring sophisticated tool-use and multi-step reasoning (Lin et al., 2025), ReQueR serves as a universal task-decomposer. Instead of the prohibitive cost of fine-tuning every specialized agent for instruction following, the Refiner reformulates global objectives into a sequence of atomic, model-invariant sub-tasks. This orchestration ensures that heterogeneous agents can execute complex workflows with high fidelity, effectively lowering the barrier for deploying advanced agentic intelligence in resource-constrained or distributed environments.

Privacy-Preserving Edge Reasoning ReQueR further facilitates a hybrid inference paradigm where data sovereignty and reasoning capacity are simultaneously optimized. A cloud-based Refiner can generate "reasoning-enhanced" meta-logical templates that are strictly independent of sensitive user content. These templates are subsequently combined with private data locally to guide a Small Language Model (SLM). Such a decoupling allows edge devices to inherit the reasoning depth of giant models (Li et al., 2024a) while ensuring that private information never leaves the local environment, defining a new frontier for secure, decentralized AI.

H Use of Large Language Models

We utilized large language models in the following aspects of this research:

- **Data Synthesis:** DeepSeek-R1 was used to generate 6,144 high-quality instruction-following samples for the cold-start Supervised Fine-Tuning phase of the Refiner model.
- **Manuscript Preparation:** Gemini was employed for grammar and spell checking to improve the clarity and readability of the manuscript.

Table 21: Comprehensive taxonomy of emergent strategies for converting human intuition into machine-executable logic.

Strategy 1: Symbolic Parameterization

Description: Transforms narrative word problems into formal symbolic models comprising variables and equations.

	Raw Query	Refined Query
Ex.	Ten treeks weigh as much as three squigs and one goolee. Two treeks and one goolee are equal in weight to one squig. The combined weight of how many treeks equals the weight of one squig?	**Let T be the weight of one treek, S be the weight of one squig, and G be the weight of one goolee.** We are given: **1.** $10T = 3S + G$, **2.** $2T + G = S$. We need to **find the number** n **such that** $nT = S$. Determine n by solving the system of equations.

Strategy 2: Contextual De-noising

Description: Filters out factual but computationally irrelevant narrative noise from the problem description.

	Raw Query	Refined Query
Ex.	~~Daniel works at an electronics store, and he claims that~~ the popularity of a television is inversely proportional to its cost. If 15 customers buy a television that costs $1500, how many customers would buy a television that costs $2500?	**Given that** the number of customers buying a television is inversely proportional to its cost, and that 15 customers buy a television costing $1500, how many customers would buy a television costing $2500?

Strategy 3: Linguistic Disambiguation

Description: Resolves semantic ambiguity by replacing vague natural language with formal operators or explicit sets.

	Raw Query	Refined Query
Ex.	What is the sum of all of the multiples of 3 between 100 and 200 ?	Calculate the sum of the **arithmetic series** starting at **102**, ending at **198**, with a **common difference of 3**.

Strategy 4: Conceptual Reduction

Description: Decomposes high-level math concepts into primitive algebraic operations and decision constraints.

	Raw Query	Refined Query
Ex.	How many vertical asymptotes does the graph of $y = \frac{2}{x^2+x-6}$ have?	Find the number of **distinct real numbers** x for which the **denominator** of the function $y = \frac{2}{x^2+x-6}$ **is zero**.

Strategy 5: Procedural Hint Injection

Description: Augments the query with necessary domain formulas or constants to minimize Solver-side retrieval failures.

	Raw Query	Refined Query
Ex.	Evaluate $i^5 + i^{-25} + i^{45}$	Evaluate the expression $i^5 + i^{-25} + i^{45}$ by **simplifying each term using the cyclical pattern** of the powers of the imaginary unit i, which repeats every four exponents: $i^0 = 1, i^1 = i, i^2 = -1, i^3 = -i$.

Strategy 6: Explicit Grounding

Description: Elevates implicit mathematical axioms (e.g., non-zero denominators) to mandatory directives.

	Raw Query	Refined Query
Ex.	For what value of x will $\frac{2x-1}{2x+2}$ and $\frac{x-3}{x-1}$ be equal?	Solve the equation $\frac{2x-1}{2x+2} = \frac{x-3}{x-1}$ for x, **ensuring to exclude any values that make the denominators zero**. Find the value of x that satisfies this equation.

Strategy 7: Chain-of-Thought Structuring

Description: Enforces a sequential reasoning path via step-by-step instructions, converting latent reasoning into explicit execution.

	Raw Query	Refined Query
Ex.	Given the functions $f(x) = \frac{x+5}{3}$ and $g(x) = \frac{1}{f^{-1}(x)+1}$, find the value of $g(3)$.	Given the function $f(x) = \frac{x+5}{3}$, **first determine its inverse function** $f^{-1}(x)$. Then, **define a new function** $g(x) = \frac{1}{f^{-1}(x)+1}$. **Finally, compute the value of** $g(3)$.

Table 22: Cross-Domain Generalization Case Studies in MMLU-Pro: ReQueR Transforming Human Queries into Structured Logic.

Domain: Law (De-noising & Conceptual Reduction)

Mechanism: Filters personal anecdotes (editor's daughter) and maps the dispute to the constitutional intersection of Commercial Speech vs. Private Editorial Control.

	Raw Query)	Refined Query
Ex.	(Narrative about a newspaper editor refusing a test-prep ad because his daughter failed the course and now works at the mall)... In an action to require the newspaper to publish, who will prevail?	Analyze whether a private newspaper can be compelled to publish an advertisement under the **First Amendment commercial speech doctrine**. Evaluate the **private actor's editorial discretion** against the owner's claim for access, considering that no state action is involved.

Domain: Biology (Linguistic Disambiguation & CoT Structuring)

Mechanism: Converts a general "Explain" prompt into a rigorous experimental design task with a clear methodological sequence.

	Raw Query	Refined Query
Ex.	Using conditioning, how can one determine the sensitivity of a dog's hearing to sounds of different frequencies?	**Design an experimental procedure using classical conditioning** to measure hearing sensitivity. **First, describe the pairing** of specific frequencies with a stimulus. **Then, outline how to test sensitivity limits** by varying frequency while holding other environmental variables constant.

Domain: Business (Symbolic Parameterization & CoT Structuring)

Mechanism: Automatically parameterizes supply and demand functions, converting a market narrative into a solvable system of equations.

	Raw Query	Refined Query
Ex.	Distributors are willing to buy $60 - p$ barrels, while the Republic is willing to supply $p^2/10$. i) What is the equilibrium price? ii) How many barrels will be sold?	**Let** $D(p) = 60 - p$ **and** $S(p) = p^2/10$. **First, find the equilibrium price** p by setting the supply and demand functions equal ($S(p) = D(p)$). **Then, evaluate the equilibrium quantity** by substituting p back into either function.

Domain: Engineering (Conceptual Reduction & Hint Injection)

Mechanism: Bridges the gap between a physical property description and a mathematical operator ($\frac{Z_L - Z_0}{Z_L + Z_0}$).

	Raw Query	Refined Query
Ex.	Find the reflection coefficient for voltage waves at the load end of a 50 ohm transmission line terminated with a load impedance of $25 - j75$ ohms.	Given characteristic impedance $Z_0 = 50\Omega$ and load impedance $Z_L = 25 - j75\Omega$, compute the reflection coefficient Γ **using the formula** $\Gamma = (Z_L - Z_0)/(Z_L + Z_0)$. Express the result in polar form.

Domain: Computer Science (Explicit Grounding)

Mechanism: Identifies the specific logical failure mode (post-target iteration) to ground the debugging task in precise algorithmic behavior.

	Raw Query	Refined Query
Ex.	Which of the following procedure calls can be used to demonstrate that the procedure does NOT Work as intended? (Followed by pseudocode)	The procedure contains a logic flaw where **foundIndex is reset to 0 if a non-target name is encountered after the match**. Identify a test case that **specifically triggers this failure** by placing the target name in a non-terminal position in the list.

Domain: Economics (Boundary Limitation)

Mechanism: Explicitly incorporates the "ceteris paribus" condition (keeping other factors constant) to satisfy formal economic definitions and logical boundaries.

	Raw Query	Refined Query
Ex.	What is meant by the marginal product of a productive factor?	Define the marginal product of a productive factor in economics, specifying that it refers to the additional output produced by an additional unit of that factor **while keeping all other factors constant**.

Domain: Physics (Hint Injection)

Mechanism: Proactively supplies missing physical constants (e.g., gravitational acceleration) when handling imperial units to eliminate Solver uncertainty.

	Raw Query	Refined Query
Ex.	In what distance can a 3000-lb automobile be stopped from a speed of 30 mi/hr (44 ft/sec) if the coefficient of friction is 0.70?	Given a car with a weight of 3000 lb and an initial speed of 44 ft/s, calculate the stopping distance. **Assume standard gravitational acceleration of** 32 **ft/s**^{2} to resolve unit-based ambiguity.