Tailored Primitive Initialization is the Secret Key to Reinforcement Learning

Yihang Yao[1,3], Guangtao Zeng[2], Raina Wu[2], Yang Zhang[3], Ding Zhao[1], Zhang-Wei Hong[2,3], Chuang Gan[3]
[1]CMU, [2]MIT, [3]MIT-IBM Watson AI Lab
Correspondence: yihangya@andrew.cmu.edu

Abstract

Reinforcement learning (RL) has emerged as a powerful paradigm for improving the reasoning capabilities of large language models (LLMs). Despite its success, RL faces fundamental challenges, including low sample efficiency and a strong dependence on the quality of the base model: while some models improve rapidly with limited RL updates, others require substantial training data to achieve meaningful gains. Recent studies suggest that the patterns of thinking tokens play a critical role in RL performance, and that supervised fine-tuning (SFT) on datasets exhibiting desirable reasoning patterns can reduce reliance on base models and better prepare LLMs for RL. However, how to automatically discover such patterns across tasks remains unclear. In this work, we describe thinking token patterns with reasoning primitives and argue that initializing LLMs with diverse, high-quality primitives is crucial for stable and efficient RL training. We propose Tailor, a pipeline that automatically discovers such reasoning primitives and curates SFT datasets to prepare LLMs for RL. Extensive experiments on mathematical and logical reasoning benchmarks demonstrate that Tailor consistently improves downstream RL performance, outperforming strong baselines, including methods with expert domain knowledge.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable reasoning capabilities across a broad range of application domains, including mathematical problem solving (Yu et al., 2023a; Yue et al., 2025b; Zeng et al., 2025b; Shen et al., 2025b), code generation (Xia et al., 2024; Yang et al., 2024b), and complex decision-making in agentic tasks such as API calling (Liu et al., 2024c; Prabhakar et al., 2025), autonomous driving (Li et al., 2024; Wei et al., 2024), and robotics (Yu et al., 2023b; Team et al., 2025a). To enhance

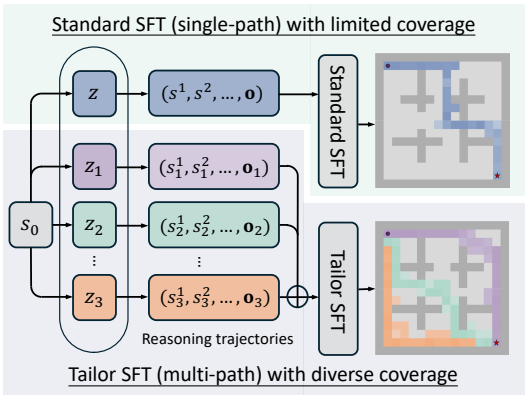


Figure 1: Reasoning token comparison. The goal in the maze refers to the correct answer, and the trajectories refer to the thinking tokens.

LLM reasoning abilities, reinforcement learning (RL) emerges as a promising paradigm by exploiting feedback from environments and leveraging verifiable reward signals (Ouyang et al., 2022; Wen et al., 2025). Such training approaches have been shown to improve model performance through the generation of long chains of thought (CoTs) (Wei et al., 2022) and test-time scaling (Yu et al., 2025c), thereby producing outputs of higher quality.

Although RL has demonstrated strong capabilities, it continues to face significant challenges. First, RL suffers from low sample efficiency (Haarnoja et al., 2018; Du et al., 2019; Shi and Chi, 2024), a limitation that is further exacerbated in the context of LLMs due to their large parameter space and the high computational cost of policy rollouts and updates (Sun et al., 2025; Zheng et al., 2025). Second, empirical studies (Gandhi et al., 2025) show that different LLMs respond inconsistently to RL: while some models exhibit substantial performance gains, others show minimal or no improvement even when trained with the same RL algorithm and data. This inconsistency suggests that successful RL training for LLMs is highly sensitive to the choice of the base model (Gandhi et al.,

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 33300–33318
July 2-7, 2026

2025; Yue et al., 2025a; Cen et al., 2025; Chen et al., 2025c). To address the challenge of low sample efficiency, one line of research focuses on improving RL algorithms through dynamic sampling (Yu et al., 2025c) and data selection techniques (Zheng et al., 2025; Sun et al., 2025). Another line of work adopts a data-centric perspective (Yu et al., 2025a) to mitigate sensitivity to base models. Prior efforts (Gandhi et al., 2025; Cen et al., 2025) have shown that injecting emergent reasoning behaviors, such as reflection, into LLMs via supervised fine-tuning (SFT) before RL can substantially improve training efficiency. However, these reasoning patterns are typically identified within specific domains, and how to generalize them to tasks with limited prior knowledge remains unclear.

In this work, we propose the **T**ask-**A**dapt**I**ve, **L**earning-**P**rimitive-**O**riented **R**einforcement (Tailor) fine-tuning pipeline, which automatically discovers diverse reasoning primitives to initialize LLMs for subsequent RL training. Specifically, a reasoning *primitive* refers to a recurring pattern in thinking tokens (e.g., bottom-up reasoning, top-down reasoning), as illustrated in Figure 3. Rule-based traces or distilled target behaviors from prior work (Gandhi et al., 2025; Cen et al., 2025) often exhibit limited coverage of the reasoning token distribution, as shown in Figure 1. This limited coverage leads to inefficient exploration and slower downstream improvement during RL. To address this limitation, the proposed Tailor pipeline automatically discovers reasoning primitives across different tasks and base models with broader coverage, thereby enabling more efficient exploration and stronger performance gains during the RL stage. Our contributions are summarized as follows:

1. Analysis of reasoning primitives. We investigate the role of reasoning primitives in warm-start RL for LLMs, shifting the focus from hand-crafted reasoning styles to more general and broader patterns, and emphasizing the importance of primitive quality and diversity.

2. Tailor pipeline. We introduce the Tailor pipeline, which automatically discovers primitives and curates SFT datasets to better initialize LLMs for subsequent RL across different tasks.

3. Comprehensive experiments. We evaluate Tailor on mathematical and logical reasoning tasks across multiple base models, demonstrating its strong potential for improving RL efficiency and its compatibility with different RL algorithms.

2 Preliminary

Markov Decision Process (MDP). We model the reasoning and action process of a large language model (LLM) under reinforcement learning as a Markov Decision Process (MDP) (Puterman, 2014), defined by the tuple $M = (\mathcal{S}, \mathcal{A}, P, r, s_0)$. Here, $\mathcal{S}$ denotes the state space, where each state $s \in \mathcal{S}$ encodes the current reasoning context of the LLM, including the history of reasoning tokens. The initial state $s_0 \in \mathcal{S}_0$ corresponds to a query from the query set $\mathcal{S}_0$. The action space $\mathcal{A}$ consists of all possible reasoning steps, where an action $a \in \mathcal{A}$ represents either an intermediate reasoning token or the final answer. The transition function is deterministic and defined as: $P(s_{t+1} \mid s_t, a_t) = \mathbb{I}[s_{t+1} = [s_t, a_t]]$, where each new state is formed by appending the chosen action to the current state. The reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ specifies the immediate reward received upon taking action a in state s. A policy $\pi_\theta : \mathcal{S} \to \mathcal{A}$ maps each state to a probability distribution over actions. A trajectory τ is defined as a sequence of states $\tau = (s_1, \ldots, s_t, ..., s_{T-1}, \mathbf{o})$, where s_1 through s_{T-1} represent intermediate reasoning steps (thinking tokens) and $\mathbf{o}$ is the final answer. T denotes the number of steps.

Reinforcement Learning. The goal of Reinforcement Learning in LLMs is to optimize the expected reward over a set of queries $\mathcal{S}_0$:

$$\max_\theta \mathcal{J} = \mathbb{E}_{s_0 \sim \mathcal{S}_0,\, s \sim \pi_\theta} \left[\sum_{t=1}^{T} r(s_0, \tau^{(\leq t)}) \right] \quad (1)$$

where $r(s_0, \tau) = \mathbf{1}(\mathbf{o} = \mathbf{o}_{\text{gold}})$, $\mathbf{o}_{\text{gold}}$ is the ground-truth answer to the query. In this work, we study RL with a rule-based outcome reward model (ORM) that assigns a binary signal to the final answer tokens in the generated output. We study the DAPO/-GRPO (Yu et al., 2025c; Shao et al., 2024)-style RL algorithms. Please see the appendix for details.

Warm-Start RL. This work focuses on the warm-start RL pipeline, which first applies Supervised Fine-Tuning (SFT) followed by RL training (Shao et al., 2024). The SFT stage enables LLMs to follow formatting instructions, become familiar with the dataset, and acquire initial reasoning capabilities within the target domain using a demonstration dataset $\mathcal{D}_{\text{SFT}}$. During SFT, the policy π_θ is trained by minimizing the negative log-likelihood loss For simplicity, we use the term SFT model to refer to the model after SFT. Related works are discussed in Appendix C.

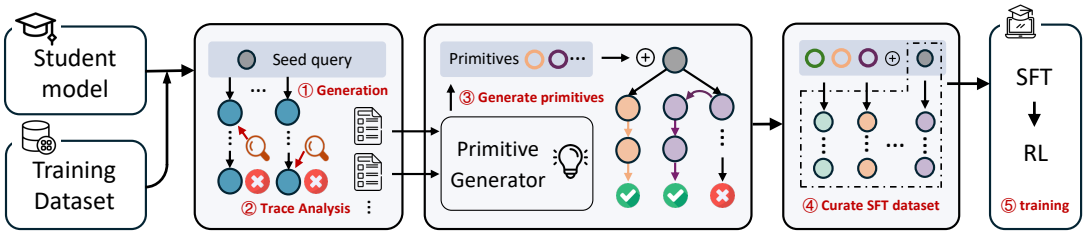


Figure 2: Overview of the Tailor finetuning pipeline.

3 Method

3.1 Reasoning Primitive

In warm-start RL, LLMs initially adopt the thinking token distribution from the SFT dataset and progressively refine their reasoning patterns in RL through interaction with verifiers. Empirical studies (Gandhi et al., 2025; Cen et al., 2025) show that the distribution of thinking tokens in the SFT dataset strongly influences downstream reasoning performance: when warm-starting from two SFT datasets with different thinking token patterns, the resulting RL performance can diverge dramatically, even though both patterns are valid, self-contained, and interpretable to humans. To investigate this phenomenon and identify better patterns, we introduce the notion of *primitives* z, which describe the trajectory-wise distribution ρ of thinking tokens. Formally, we augment the MDP tuple $M' = (\mathcal{S}, \mathcal{A}, P, r, s_0, z)$ with a reasoning primitive z (Qu et al., 2025):

$$\rho(\pi_\theta \mid z) = \mathbb{E}_{s_0 \sim \mathcal{S}_0} \prod_t \pi_\theta(a_t \mid s_t, z)\, \mathbb{I}[s_{t+1} = [s_t, a_t]] \quad (2)$$

In the context of LLMs, primitives z can be instantiated as prompt instructions that are combined with the query s_0 to guide the teacher model's reasoning and generate the SFT dataset. We provide several textual examples of reasoning primitives in Figure 3. For example, when constructing reasoning chains for a math problem, different primitives such as *top-down* and *bottom-up* reasoning can be applied. Primitives also encompass broader behaviors, such as *reflection* and *backtracking*, which enable the model to detect and recover from failures during the reasoning process. This concept of primitives naturally extends beyond mathematical reasoning to other domains, such as software engineering tasks (Zhang et al., 2025), where primitives arise from variations in agentic designs and prompt structures.

Math question: The number of each Manager's Backpack equals the difference between ... How many Manager Backpack does each Graphic Design Studio have?

Primitive 1: Top-Down
1) Dependency chain discover
2) Problem breakdown
⋮
7) Information gathering
8) Final answer derivation

Primitive 2: Bottom-Up
1) Variable Definition
2) Build Equations
3) Solve intermediate results
⋮
7) Final answer derivation

Figure 3: Examples of reasoning primitives.

3.2 Tailor Pipeline

Primitive initialization plays a critical role in warm-start RL. Due to the strong regularization imposed by RL and the vast space of natural language, LLMs often struggle to autonomously discover novel reasoning primitives during the RL process (Zhao et al., 2025), especially when training specialized LLMs with limited model capacity or restricted pre-training data distributions. Prior work has identified specific reasoning patterns, including *backtracking*, *reflection*, *setting subgoals*, and *backward chaining* (Gandhi et al., 2025), that lead to improved performance in subsequent RL training. We interpret the effectiveness of these primitives through two key requirements: (1) **Quality**: primitives such as *setting subgoals* and *backward chaining* provide domain-appropriate inductive biases that directly facilitate problem solving; and (2) **Diversity**: primitives such as *backtracking* and *reflection* introduce stochasticity into the decoding of thinking tokens, thereby increasing uncertainty over reasoning trajectories and implicitly encouraging exploration of alternative reasoning paths.

Primitive Quality. As noted by (Qu et al., 2025), high-quality primitives enable LLMs to achieve superior performance on specific problem sets. We interpret primitive quality along two dimensions: *task-specificness* and *student-model awareness*. Task-specificness requires reasoning

patterns that are correct, natural, and well aligned with the target domain, enabling LLMs to reliably generate correct solutions. Student-model awareness requires that the primitives used in the SFT dataset do not deviate excessively from the student model's pre-training data distribution, echoing insights from Generalized Knowledge Distillation (Agarwal et al., 2024). An SFT dataset with limited distributional shift helps LLMs adapt to target domains while preserving exploration capability and reducing the risk of overfitting.

Primitive Diversity. Optimal reasoning primitives for a given domain are often not unique: agents initialized with different prompts can successfully solve distinct subsets of software engineering tasks (Zhang et al., 2025). This observation highlights the importance of primitive diversity. Moreover, identifying a single optimal high-quality primitive is particularly challenging in novel or low-prior tasks. Increasing primitive diversity, therefore, expands the coverage of thinking tokens within the target primitive space, facilitating more robust exploration and downstream RL. In our work, thinking token (tokens inside <think>...</think>) diversity and coverage are operationalized as the support size of the reasoning trajectory distribution over thinking tokens. We operationalize coverage via two measurable proxies: (1) the number of reasoning primitives ($|z|$) used during SFT curation; (2) the semantic dispersion of thinking tokens using embedding models. More details are provided in the experiments.

To meet the objectives of diversity and quality, we propose the **T**ask-**A**dapt**I**ve, **L**earning-**P**rimitive-**O**riented **R**einforcement (Tailor) fine-tuning pipeline, illustrated in Figure 2. It automatically discovers diverse primitives with high quality and prepares student models ready for reasoning tasks. Key components of Tailor are as follows:

Demonstration Trace Analysis. The first step prompts the instructed student LLMs to generate completions on a small subset of the training data, with answers labeled using verifiers. These demonstrations contain thinking tokens that remain close to the student models' pre-training distributions while exhibiting a wide range of failures and errors, as student models typically perform poorly in the target domains. We then employ a teacher model[1] to analyze these traces and, upon detecting failures, propose corrections and alternative reasoning paths toward the correct answers. This step is designed to uncover diverse, repair-oriented reasoning patterns that stay close to the student models' pre-training distributions, thereby making them easier for the student models to adopt and follow.

[1]We deploy DeepSeek-V3-0324 (Liu et al., 2024b) as the teacher model in our experiments.

Algorithm 1 Tailor Finetuning Pipeline

Input: SFT seed dataset $\mathcal{D}_{\text{s}}$, subsets of RL dataset $\mathcal{D}_{\text{RL}}$, student model π_θ, teacher model π_t.
Output: Tailor RL model $\pi_{\theta'}$.

```
# Demonstration trace analysis.
Generate completions:
  τ ~ π_θ(· | s_0), s_0 ~ D_RL
Summarize Failures F = {f_k}, f_k ← π_t(τ_k).
# Reasoning primitive synthesis.
Generate Z = {z_m}: z_m ← π_t(f_m)
# Tailor SFT dataset curation.
Initialize SFT D_SFT ← ∅
for (s_0, a) ∈ D_s do
    Sample a primitive z ~ Z;
    Curate trace: τ ← π_t(· | s_0, z);
    Update D_SFT ← D_SFT ∪ (s_0, τ)
end for
# SFT & RL.
Perform SFT on D_SFT and RL on D_RL.
Return: RL policy π_θ'
```

Reasoning Primitive Synthesis. Building on the summarized traces from the previous step, we synthesize reasoning primitives by prompting the teacher model to generate corresponding primitives z that guide LLMs to produce failure-recovery reasoning patterns during generation. Given that the observed failures span a wide range of types, we are able to curate a broad set of reasoning instructions to support the subsequent SFT dataset curation.

Tailor SFT Dataset Curation and Training. After obtaining the set of reasoning primitives Z, we prompt the LLM teacher model to generate reasoning traces that explicitly follow each primitive. With the Tailor SFT dataset, we fine-tune the student models on it and subsequently apply RL. A simplified overview of the Tailor process is provided in Algorithm 1, and additional details are included in the Appendix A.

Intuition. In the Tailor pipeline, student models often exhibit a variety of reasoning failures in their demonstration traces. These failures include examples such as skipping intermediate steps, making unsupported assumptions, or mishandling mathematical relationships. By analyzing these

traces, the teacher model constructs a set of primitives $\mathcal{Z}$ that directly address the observed error patterns. This set is not only tailored to the student model's generation behaviors and the target domain (**Quality requirements**) but is also enriched with diverse instructions that correspond to distinct types of reasoning errors. (**Diversity requirements**). Consequently, Tailor can lead to more efficient RL and shows greater performance gains.

4 Experiments

4.1 Experiment Setup

Dataset and Benchmarks. We conduct experiments on two reasoning benchmarks: (1) iGSM (Ye et al., 2024), a grade-school math benchmark that tests mathematical and commonsense reasoning skills, containing difficulty of medium and hard tasks; and (2) KK (Knights & Knaves) (Xie et al., 2024), a logical reasoning benchmark based on dynamically generated knights and knaves puzzles. We choose these two benchmarks to assess reasoning capabilities in LLMs while reducing the risk of data contamination from the pre-training stage (Wu et al., 2025), as the synthetic nature of iGSM and KK helps mitigate such issues (Ye et al., 2024, 2025). For both datasets, we evaluate methods in in-distribution (In-Dist) data and out-of-distribution (OOD) data.

For the KK task, we simulate a practical setting in which the teacher model performs well, while the student model fails to achieve satisfactory performance. The teacher model achieves approximately 95% accuracy on the SFT dataset. We do not perform rejection sampling in this setting, as incorrect answers are rare and still provide useful learning signals (Xie et al., 2024). In contrast, for the iGSM tasks, we simulate a scenario where the teacher model performs poorly, achieving only around 25% accuracy on the SFT dataset. In this case, we apply rejection sampling during SFT data collection to filter out incorrect answers, while keeping the overall dataset size unchanged. See Appendix A for more details.

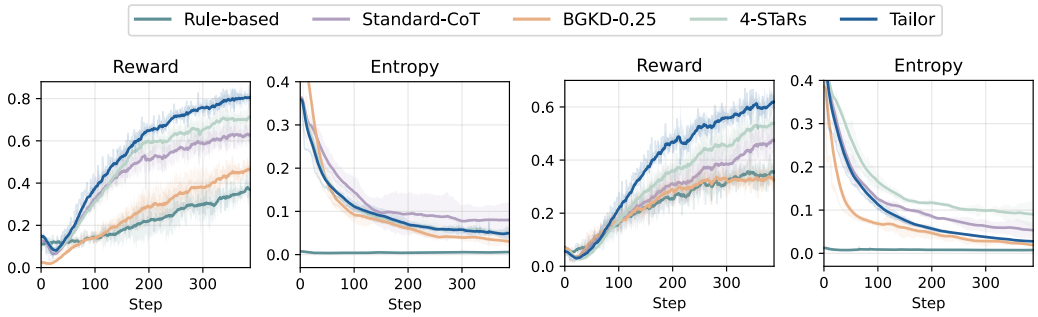
Baselines. We compare Tailor with three types of baselines: (1) Rule-based Demonstration. Both the iGSM and KK datasets provide ground-truth functions to generate rule-based, self-contained reasoning traces with expert knowledge. We perform SFT on the data, and then apply RL. We refer to this baseline as **`Rule-based`**. (2) Human-Crafted Primitives. Gandhi et al. (2025) identifies four critical STaR behaviors for RL and injects these reasoning patterns by prompting a teacher model with specialized instructions. We use their prompts to collect demonstrations and perform distillation, referring to this baseline as **`4-STaR`**. Additionally, we include a **`Standard-CoT`** baseline in this category, where teacher demonstrations are generated using CoT prompting (Wei et al., 2022). (3) Re-distillation. Chen et al. (2025c) propose a re-distillation strategy in which a trained model is used to regenerate the SFT dataset, producing data that better aligns with the model's pre-training distribution. This idea aligns with Generalized Knowledge Distillation (GKD) (Agarwal et al., 2024), which aims to reduce the distributional gap between the finetuning dataset and the student model's pre-training distribution. We apply re-distillation to the 4-STaR SFT model to curate a new dataset, which we then combine with the original 4-STaR dataset. After SFT on this mixed data, we apply RL. We refer to this baseline as Batch-GKD (**`BGKD-`**λ), where $\lambda \in [0, 1]$ denotes the proportion of re-distilled data in the SFT dataset.

Models. We evaluate our method using the Llama (Grattafiori et al., 2024) and Qwen (Yang et al., 2024a) model families. For Llama, we use Llama3.2-1B and Llama3.2-3B as base models; for Qwen, we use Qwen2.5-0.5B and Qwen2.5-3B. Unless otherwise specified, we use DeepSeek-V3 (Liu et al., 2024a) as the teacher model with a decoding temperature of 0.5. The evaluation temperature for both SFT and RL models is set to 1.0. Additional details are provided in Appendix A.

4.2 Main Results

We present the RL performance on the KK and iGSM tasks in Figure 5 and 6, respectively. We can observe that both Llama and Qwen models struggle to achieve competitive performance after RL with rule-based CoTs. This suggests that accuracy and reasoning consistency alone are insufficient indicators of whether an SFT dataset provides good demonstrations for efficient RL. For example, rule-based annotations yield perfectly accurate and coherent reasoning traces, yet still fail to support effective RL training. These findings highlight the importance of curating SFT datasets that better prepare LLMs for RL.

Re-distillation methods (BGKD-λ) show modest performance improvements during the SFT stage. By using self-generated data, these methods reduce the distributional gap between SFT and pre-training



(a) LLama3.2-1B as base models (b) Qwen2.5-0.5B as base models

Figure 4: Training Curves of the KK tasks. We average curves with 3 random seeds.

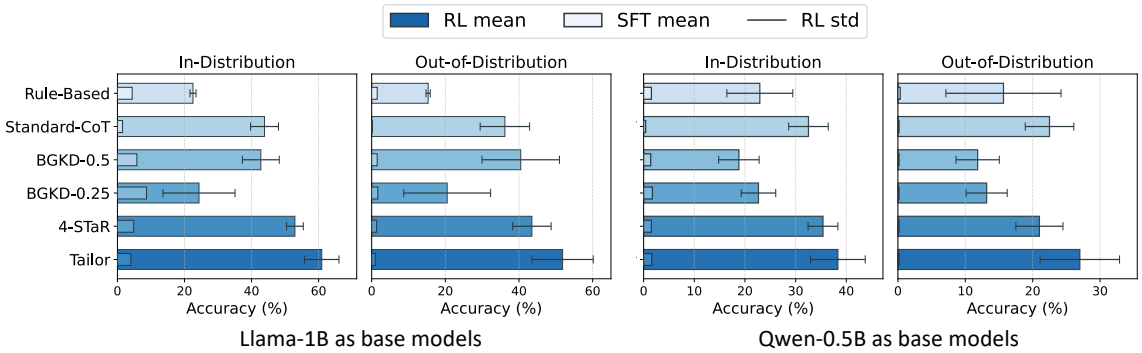


Figure 5: The evaluation results (%) on the KK reasoning tasks. We train SFT models for 4 epochs and finetune them with RL for 5 epochs. The mean value and standard deviation are calculated over 3 random seeds.

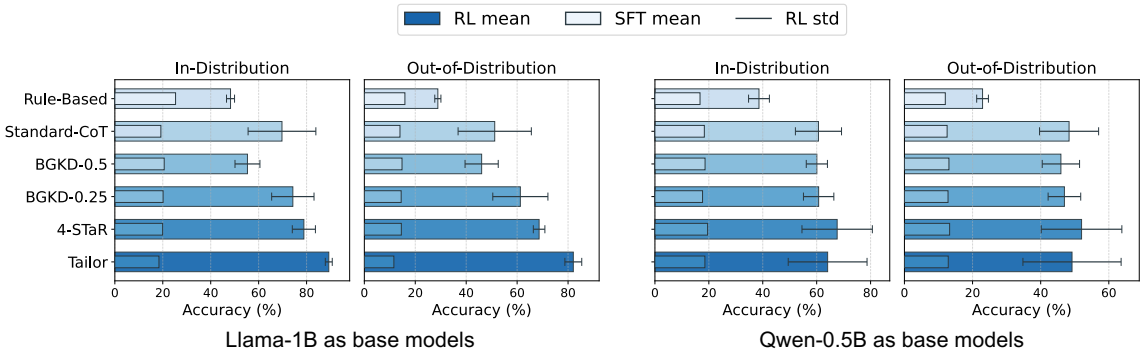
data, resulting in more learnable patterns. However, BGKD still underperforms after the RL stage, as relying solely on re-distilled data diminishes the exploration benefits provided by the teacher model. Baselines such as Standard-CoT and 4-STaR achieve substantial RL performance gains by incorporating specific reasoning behaviors from teacher demonstrations. However, 4-STaR is with strong prior knowledge and it still achieves sub-optimal performance compared to our Tailor. Finally, our method Tailor achieves the best overall RL performance and the largest improvements on both in-distribution and OOD evaluation sets, benefiting from the more diverse reasoning primitives it provides.

We should notice that the better RL performance is not directly correlated with SFT models accuracy: As shown in Figure 5 and 6. Tailor does not consistently achieves the best SFT performance. In contrast, Rule-based and BGKD sometimes achieve significantly better SFT performance. This pattern can be interpreted as Tailor pipeline curate SFT dataset with diverse primitives, and it requires the subsequent RL stage to help select suitable patterns with reward signals. Baselines may contain less incorrect reasoning patterns, but with the less diverse primitives, the ceiling of RL performance is also limited. Consequently, *SFT performance itself can not serve as the solo indicator for RL potentials.*

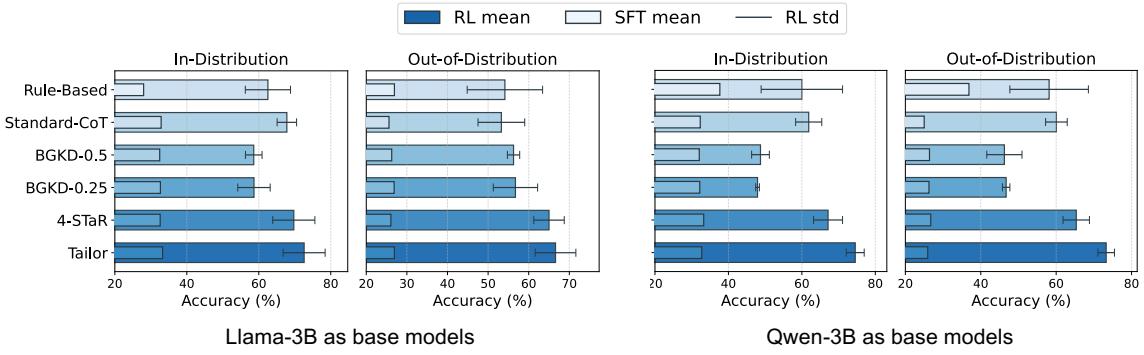
As shown in Figure 4, Tailor also exhibits faster learning and higher sampling efficiency, attributed to its higher initial thinking token coverage. The entropy of the baseline methods, Standard-CoT and 4-STaRs, also remains high, but their training rewards are lower. This suggests that *policy entropy itself can not reflect the exploration capability in RL*: models may be exploring only superficial variations, such as changing individual words, rather than engaging in meaningful strategy-level exploration. Consequently, the resulting benefit for RL is limited.

4.3 Diversity Analysis

To evaluate the diversity of reasoning primitives in the curated SFT datasets, we use NV-Embed-v2 (Lee et al., 2024a) to compute embeddings of the thinking tokens and calculate the average pairwise cosine similarity for responses to the same seed query. Results on the KK dataset are shown in Figure 8. We observe that 4-STaR improves diversity over Standard-CoT, as reflected by a lower median similarity and a longer tail extending toward lower



(a) Evaluation on the iGSM-medium task



(b) Evaluation on the iGSM-hard task

Figure 6: The evaluation results (%) on the iGSM reasoning tasks. We train SFT models for 4 epochs and finetune them with RL for 5 epochs for the iGSM-medium tasks, and finetune with RL for 100 steps for the iGSM-hard tasks. The mean value and standard deviation are calculated over 3 random seeds.

values. This improvement can be attributed to the inclusion of exploration behaviors such as *reflection* and *backtracking*, which increase variation in the generated reasoning traces. Moreover, our proposed method Tailor further reduces the similarity scores compared to 4-STaR, with an even lower median and a heavier tail in the low-similarity region. This indicates that Tailor significantly enhances high-level reasoning diversity in the SFT dataset, better preparing LLMs for effective exploration during RL.

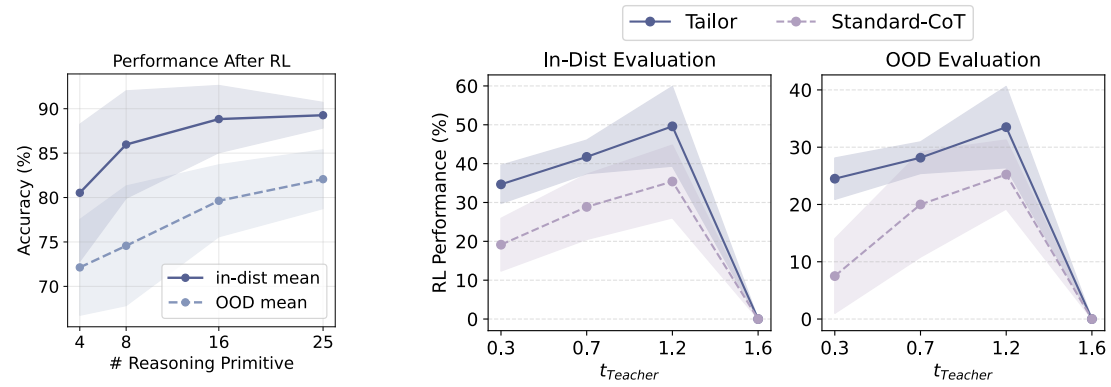
Takeaways of Section 4.2 and 4.3

(1) High pass@1 rate of SFT checkpoints and high policy entropy alone do not guarantee high performance after RL.
(2) Primitive coverage, instead of wording diversity, is crucial for model initialization.
(3) Tailor prepares models with diverse high-quality primitives, thus achieving top performances among strong baselines.

Ablation on Reasoning Primitives. In our main experiments, we use 25 reasoning primitives for SFT data collection. To investigate the effect of diversity and coverage in reasoning primitives, we vary the size of the primitive set from 4 to 25 while keeping the overall SFT dataset size fixed. We perform ablations on the iGSM-medium task using Llama-1B as the base model. The results are shown in Figure 7 (a). We observe that when the primitive set is small and less diverse, the resulting RL performance is lower. This is likely because the limited primitive set covers only a small subset of effective strategies, making it difficult to generalize across a wide range of questions and hard for exploration. As the number of reasoning primitives increases to 25, post-RL performance improves, highlighting the importance of primitive coverage when preparing LLMs for RL.

4.4 Ablation Study

Ablation on Teacher Model Temperature. Adjusting the teacher model's sampling temperature t_{Teacher} is a common approach for controlling the diversity and coverage of demonstration datasets (Mukherjee et al., 2023; Hong et al., 2024). We vary the teacher's decoding temperature and



(a) Ablation on Reasoning Primitive (b) Ablation on Teacher Model Decoding Temperature

Figure 7: Ablation Study. (a) Effect of the number of reasoning primitives in the SFT dataset. Experiments are conducted on the iGSM-medium dataset using Llama3.2-1B as the base model. (b) Effect of the teacher model's decoding temperature. Experiments are conducted on the KK dataset using Qwen2.5-0.5B as the base model.

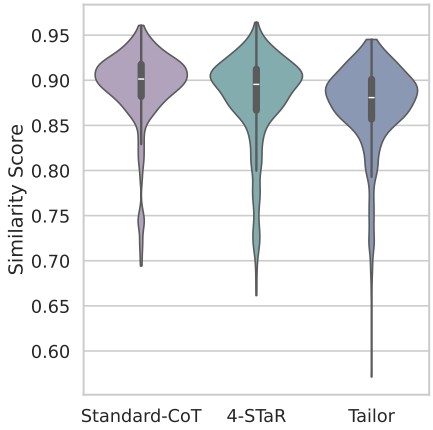


Figure 8: Primitive similarity: lower score means higher reasoning diversity.

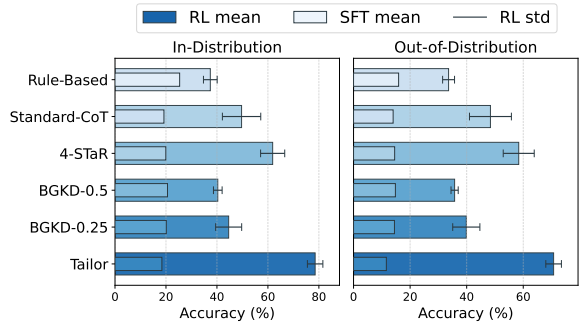


Figure 9: GRPO experiments on the iGSM-medium task with LLama-1B as base models.

conduct ablation studies on the KK task using Llama-1B as the base model. The results are shown in Figure 7(b). We observe that increasing the decoding temperature generally improves RL performance, indicating that greater diversity in teacher outputs benefits downstream training. Across all temperature settings, however, Tailor consistently outperforms the baselines, demonstrating a more effective balance between demonstration quality and diversity. Notably, when the temperature becomes excessively high (e.g., $t = 1.6$), demonstration quality degrades substantially, and the student model fails to learn meaningful reasoning traces during SFT, leading to near-zero accuracy after RL.

Ablation on Alternative RL Algorithms. In addition to DAPO, we also combine our Tailor pipeline with GRPO (Shao et al., 2024) on the iGSM-medium task using Llama-1B as the base model. The results are shown in Figure 9, where we observe that the conclusions remain consistent with those from the DAPO: Tailor achieves the highest RL improvement and final performance. These results demonstrate the compatibility of our approach with a broader range of RL algorithms.

Takeaways

(1) Primitive diversity in SFT datasets is critical for successful warm-starting of RL. **(2)** Tailor hows robust performance across hyperparameters and RL algorithms.

5 Conclusion

In this paper, we interpret the variation in RL performance across different initial models, as well as the issue of low sampling efficiency, through the lens of reasoning primitive quality and coverage. To address these challenges, we adopt a data-centric perspective and focus on constructing SFT datasets composed of high-quality and diverse reasoning primitives. We propose Tailor, a pipeline that automatically discovers diverse, high-quality reasoning primitives and constructs demonstration data to warm-start LLMs for RL. By increasing the

coverage of the thinking-token distribution, Tailor enables faster exploration and unlocks greater performance potential during RL training. Extensive experiments across multiple benchmarks demonstrate that Tailor effectively curates a more diverse training corpus and substantially improves downstream RL performance. A potential negative impact is that misuse of our method could lead to the generation of unsafe or toxic primitives.

Limitations

One limitation of Tailor is current evaluation focuses on verifiable reasoning tasks and single-turn RL. As future work, we plan to extend the pipeline to broader domains. Nevertheless, we believe Tailor sheds light on data-centric RL, emphasizing that a successful RL process depends on well-initialized models with diverse and high-quality thinking trajectory distribution.

Acknowledgment

We would like to thank the anonymous reviewers and the decision committee for their review efforts and constructive feedback.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. 2024. On-policy distillation of language models: Learning from self-generated mistakes. In *The twelfth international conference on learning representations*.

Zhepeng Cen, Yihang Yao, William Han, Zuxin Liu, and Ding Zhao. 2025. Behavior injection: Preparing language models for reinforcement learning. *arXiv preprint arXiv:2505.18917*.

Kevin Chen, Marco Cusumano-Towner, Brody Huval, Aleksei Petrenko, Jackson Hamburger, Vladlen Koltun, and Philipp Krähenbühl. 2025a. Reinforcement learning for long-horizon interactive llm agents. *arXiv preprint arXiv:2502.01600*.

Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Fan Yang, Zenan Zhou, Weipeng Chen, Haofen Wang, Jeff Z Pan, and 1 others. 2025b. Learning to reason with search for llms via reinforcement learning. *arXiv preprint arXiv:2503.19470*.

Yutong Chen, Jiandong Gao, and Ji Wu. 2025c. Towards revealing the effectiveness of small-scale fine-tuning in r1-style reinforcement learning. *arXiv preprint arXiv:2505.17988*.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. 2025. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*.

Simon S Du, Sham M Kakade, Ruosong Wang, and Lin F Yang. 2019. Is a good representation sufficient for sample efficient reinforcement learning? *arXiv preprint arXiv:1910.03016*.

Xidong Feng, Ziyu Wan, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. 2023. Alphazero-like tree-search can guide large language model decoding and training. *arXiv preprint arXiv:2309.17179*.

Yuqian Fu, Tinghong Chen, Jiajun Chai, Xihuai Wang, Songjun Tu, Guojun Yin, Wei Lin, Qichao Zhang, Yuanheng Zhu, and Dongbin Zhao. 2025. Srft: A single-stage method with supervised and reinforcement fine-tuning for reasoning. *arXiv preprint arXiv:2506.19767*.

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

Anirudh Goyal, Shagun Sodhani, Jonathan Binas, Xue Bin Peng, Sergey Levine, and Yoshua Bengio. 2020. Reinforcement learning with competitive ensembles of information-constrained primitives. In *International Conference on Learning Representations*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhishek Pandey, Ankur Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 11 others. 2024. The llama 3 herd of models. arXiv preprint arXiv:2407.21783. "The Llama 3 Herd of Models".

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, and 1 others. 2025. Openthoughts: Data recipes for reasoning models. *arXiv preprint arXiv:2506.04178*.

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*.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and 1 others. 2018. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*.

Lixuan He, Jie Feng, and Yong Li. 2025. Amft: Aligning llm reasoners by meta-learning the optimal imitation-exploration balance. *arXiv preprint arXiv:2508.06944*.

Zhang-Wei Hong, Pulkit Agrawal, Rémi Tachet des Combes, and Romain Laroche. 2023. Harnessing mixed offline reinforcement learning datasets via trajectory weighting. *arXiv preprint arXiv:2306.13085*.

Zhang-Wei Hong, Idan Shenfeld, Tsun-Hsuan Wang, Yung-Sung Chuang, Aldo Pareja, James Glass, Akash Srivastava, and Pulkit Agrawal. 2024. Curiosity-driven red-teaming for large language models. *arXiv preprint arXiv:2402.19464*.

Jian Hu. 2025. Reinforce++: A simple and efficient approach for aligning large language models. *arXiv preprint arXiv:2501.03262*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*.

Joongwon Kim, Anirudh Goyal, Liang Tan, Hannaneh Hajishirzi, Srinivasan Iyer, and Tianlu Wang. 2025. Astro: Teaching language models to reason by reflecting and backtracking in-context. *arXiv preprint arXiv:2507.00417*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024a. Nv-embed: Improved techniques for training llms as generalist embedding models. *arXiv preprint arXiv:2405.17428*. Includes NV-Embed-v2; ranks No. 1 on the Massive Text Embedding Benchmark (MTEB) as of August 30, 2024.

Jaewoo Lee, Sujin Yun, Taeyoung Yun, and Jinkyoo Park. 2024b. Gta: Generative trajectory augmentation with guidance for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 37:56766–56801.

Boyi Li, Yue Wang, Jiageng Mao, Boris Ivanovic, Sushant Veer, Karen Leung, and Marco Pavone. 2024. Driving everywhere with large language model policy adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14948–14957.

Yang Li, Youssef Emad, Karthik Padthe, Jack Lanchantin, Weizhe Yuan, Thao Nguyen, Jason Weston, Shang-Wen Li, Dong Wang, Ilia Kulikov, and 1 others. 2025. Naturalthoughts: Selecting and distilling reasoning traces for general reasoning tasks. *arXiv preprint arXiv:2507.01921*.

Yiqing Liang, Jielin Qiu, Wenhao Ding, Zuxin Liu, James Tompkin, Mengdi Xu, Mengzhou Xia, Zhengzhong Tu, Laixi Shi, and Jiacheng Zhu. 2025. Modomodo: Multi-domain data mixtures for multimodal llm reinforcement learning. *arXiv preprint arXiv:2505.24871*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, and 159 others. 2024a. Deepseek-v3 technical report.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024b. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. 2025. Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models. *arXiv preprint arXiv:2505.24864*.

Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, Rithesh RN, and 1 others. 2024c. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets. *Advances in Neural Information Processing Systems*, 37:54463–54482.

Lu Ma, Hao Liang, Meiyi Qiang, Lexiang Tang, Xiaochen Ma, Zhen Hao Wong, Junbo Niu, Chengyu Shen, Runming He, Bin Cui, and 1 others. 2025. Learning what reinforcement learning can't: Interleaved online fine-tuning for hardest questions. *arXiv preprint arXiv:2506.07527*.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*.

Xuan-Phi Nguyen, Shrey Pandit, Revanth Gangi Reddy, Austin Xu, Silvio Savarese, Caiming Xiong, and Shafiq Joty. 2025. Sfr-deepresearch: Towards effective reinforcement learning for autonomously reasoning single agents. *arXiv preprint arXiv:2509.06283*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. 2024. Training software engineering agents and verifiers with swe-gym. *arXiv preprint arXiv:2412.21139*.

Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. 2019. Mcp: Learning composable hierarchical control with multiplicative compositional policies. *Advances in neural information processing systems*, 32.

Akshara Prabhakar, Zuxin Liu, Ming Zhu, Jianguo Zhang, Tulika Awalgaonkar, Shiyu Wang, Zhiwei Liu, Haolin Chen, Thai Hoang, Juan Carlos Niebles, and 1 others. 2025. Apigen-mt: Agentic pipeline for multi-turn data generation via simulated agent-human interplay. *arXiv preprint arXiv:2504.03601*.

Martin L. Puterman. 2014. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 2nd edition. John Wiley & Sons.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jiadai Sun, Shuntian Yao, and 1 others. 2024. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. *arXiv preprint arXiv:2411.02337*.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*.

Yuxiao Qu, Anikait Singh, Yoonho Lee, Amrith Setlur, Ruslan Salakhutdinov, Chelsea Finn, and Aviral Kumar. 2025. Learning to discover abstractions for llm reasoning. In *Proceedings of the AI4Math Workshop, ICML 2025*. Poster.

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

Haozhan Shen, Peng Liu, Jingcheng Li, Chunxin Fang, Yibo Ma, Jiajia Liao, Qiaoli Shen, Zilun Zhang, Kangjia Zhao, Qianqian Zhang, and 1 others. 2025a. Vlm-r1: A stable and generalizable r1-style large vision-language model. *arXiv preprint arXiv:2504.07615*.

Maohao Shen, Guangtao Zeng, Zhenting Qi, Zhang-Wei Hong, Zhenfang Chen, Wei Lu, Gregory Wornell, Subhro Das, David Cox, and Chuang Gan. 2025b. Satori: Reinforcement learning with chain-of-action-thought enhances llm reasoning via autoregressive search. *arXiv preprint arXiv:2502.02508*.

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.

Laixi Shi and Yuejie Chi. 2024. Distributionally robust model-based offline reinforcement learning with near-optimal sample complexity. *Journal of Machine Learning Research*, 25(200):1–91.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*.

Yifan Sun, Jingyan Shen, Yibin Wang, Tianyu Chen, Zhendong Wang, Mingyuan Zhou, and Huan Zhang. 2025. Improving data efficiency for llm reinforcement fine-tuning through difficulty-targeted online data selection and rollout replay. *arXiv preprint arXiv:2506.05316*.

Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, and 1 others. 2025a. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025b. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

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.

Yuxi Wei, Zi Wang, Yifan Lu, Chenxin Xu, Changxing Liu, Hao Zhao, Siheng Chen, and Yanfeng Wang. 2024. Editable scene simulation for autonomous driving via collaborative llm-agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15077–15087.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I Wang. 2025. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution. *arXiv preprint arXiv:2502.18449*.

Xumeng Wen, Zihan Liu, Shun Zheng, Zhijian Xu, Shengyu Ye, Zhirong Wu, Xiao Liang, Yang Wang, Junjie Li, Ziming Miao, and 1 others. 2025. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms. *arXiv preprint arXiv:2506.14245*.

Mingqi Wu, Zhihao Zhang, Qiaole Dong, Zhiheng Xi, Jun Zhao, Senjie Jin, Xiaoran Fan, Yuhao Zhou, Huijie Lv, Ming Zhang, and 1 others. 2025. Reasoning or memorization? unreliable results of reinforcement learning due to data contamination. *arXiv preprint arXiv:2507.10532*.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. 2024. Agentless: Demystifying llm-based software engineering agents. *arXiv preprint arXiv:2407.01489*.

Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. 2024. On memorization of large language models in logical reasoning. *arXiv preprint arXiv:2410.23123*.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. 2025. Learning to reason under off-policy guidance. *arXiv preprint arXiv:2504.14945*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2024a. Qwen2.5 technical report. arXiv preprint arXiv:2412.15115. "Qwen2.5 Technical Report".

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024b. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652.

Yihang Yao, Zhepeng Cen, Wenhao Ding, Haohong Lin, Shiqi Liu, Tingnan Zhang, Wenhao Yu, and Ding Zhao. 2024. Oasis: Conditional distribution shaping for offline safe reinforcement learning. *Advances in Neural Information Processing Systems*, 37:78451–78478.

Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. 2024. Physics of language models: Part 2.1, grade-school math and the hidden reasoning process. In *The Thirteenth International Conference on Learning Representations*.

Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. 2025. Physics of Language Models: Part 2.2, How to Learn From Mistakes on Grade-School Math Problems. In *Proceedings of the 13th International Conference on Learning Representations*, ICLR '25. Full version available at https://ssrn.com/abstract=5250631.

Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. 2025. Demystifying long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2502.03373*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023a. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Ping Yu, Jack Lanchantin, Tianlu Wang, Weizhe Yuan, Olga Golovneva, Ilia Kulikov, Sainbayar Sukhbaatar, Jason Weston, and Jing Xu. 2025a. Cot-self-instruct: Building high-quality synthetic prompts for reasoning and non-reasoning tasks. *arXiv preprint arXiv:2507.23751*.

Ping Yu, Weizhe Yuan, Olga Golovneva, Tianhao Wu, Sainbayar Sukhbaatar, Jason Weston, and Jing Xu. 2025b. Rip: Better models by survival of the fittest prompts. *arXiv preprint arXiv:2501.18578*.

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

Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montse Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, and 1 others. 2023b. Language to rewards for robotic skill synthesis. *arXiv preprint arXiv:2306.08647*.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. 2025a. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, and 1 others. 2025b. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*.

Guangtao Zeng, Maohao Shen, Delin Chen, Zhenting Qi, Subhro Das, Dan Gutfreund, David Cox, Gregory Wornell, Wei Lu, Zhang-Wei Hong, and 1 others. 2025a. Satori-swe: Evolutionary test-time scaling for sample-efficient software engineering. *arXiv preprint arXiv:2505.23604*.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. 2025b. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*.

Kexun Zhang, Weiran Yao, Zuxin Liu, Yihao Feng, Zhiwei Liu, Rithesh RN, Tian Lan, Lei Li, Renze Lou, Jiacheng Xu, and 1 others. 2025. Diversity empowers intelligence: Integrating expertise of software engineering agents. In *The Thirteenth International Conference on Learning Representations*.

Rosie Zhao, Alexandru Meterez, Sham Kakade, Cengiz Pehlevan, Samy Jelassi, and Eran Malach. 2025. Echo chamber: Rl post-training amplifies behaviors learned in pretraining. *arXiv preprint arXiv:2504.07912*.

Haizhong Zheng, Yang Zhou, Brian R Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. 2025. Act only when it pays: Efficient reinforcement learning for llm reasoning via selective rollouts. *arXiv preprint arXiv:2506.02177*.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, and 1 others. 2023. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36:55006–55021.

A More Details of Experiments

A.1 Evaluation Benchmark

In this section, we introduce the KK and iGSM benchmarks used for evaluation.

KK (Xie et al., 2024) is a logical reasoning benchmark composed of dynamically generated Knights and Knaves (KK) puzzles. In each puzzle, every character is either a Knight (who always tells the truth) or a Knave (who always lies). The puzzle presents a set of natural language statements made by the characters, referring to themselves or others. The task is to infer the identity (Knight or Knave) of each character based on these statements. Solving the puzzle requires ensuring global logical consistency: the assigned identities must be compatible with the logical implications of all statements. The input is the full puzzle text (i.e., all character statements), and the expected output is a complete assignment of identities for all characters. A prediction is considered correct only if the identities of all characters are correctly labeled. We provide an example of a query below.

Query:
A very special island is inhabited only by knights and knaves. Knights always tell the truth, and knaves always lie. You meet 7 inhabitants: Abigail, Aria, Jacob, Liam, James, Amelia, and David. In Abigail's words: "David is a knight if and only if Liam is a knight". Aria noted, "If Abigail is a knight then Liam is a knight". "Abigail is a knave" - Jacob. As Liam put it, "If Amelia is a knight then Abigail is a knight". "Jacob is a knight or Amelia is a knave," James claimed. "David is not a knight" - Amelia. David said that Jacob is a knave and David is a knight. So who is a knight and who is a knave?

Rule-based CoT:
<think>
Assume Abigail is a knight. No contradiction is found in their claim that David is a knight if and only if Liam is a knight. Assume David is a knight. No contradiction is found in their claim that Jacob is a knave and David is a knight. Jacob cannot be a knight, because this would contradict the claim of their own that Abigail is a knave. Assume Jacob is a knave. No contradiction is found in their false claim that Abigail is a knave. Assume Liam is a knight. No contradiction is found in their claim that If Amelia is a knight then Abigail is a knight. Amelia cannot be a knight, because this would contradict the claim of their own that David is not a knight. Assume Amelia is a knave. No contradiction is found in their false claim that David is not a knight. Assume Aria is a knight. No contradiction is found in their claim that If Abigail is a knight then Liam is a knight. Assume James is a knight. No contradiction is found in their claim that Jacob is a knight or Amelia is a knave. </think>
<answer>
Thus, the final answer is boxed{Abigail is a knight. David is a knight. Jacob is a knave. Liam is a knight. Amelia is a knave. Aria is a knight. James is a knight.}
</answer>

iGSM (Ye et al., 2024) is an infinite grade-school math problem benchmark designed to evaluate LLMs' mathematical and commonsense reasoning abilities. We apply slight modifications to iGSM to improve fine-tuning stability, including the removal of modulo operations (Cen et al., 2025). This is motivated by prior findings that base models struggle to compute modulo with high accuracy, and such capability cannot be substantially improved through training on a small SFT dataset (Ye et al., 2024). The iGSM benchmark is constructed under the assumption that ground-truth reasoning chains follow a Directed Acyclic Graph (DAG) structure. It is important to note that our proposed Tailor does not leverage any prior knowledge of this DAG representation. Implementation details, including definitions of nodes and edges, their generation process, and the natural language translation procedure, are thoroughly introduced in the original paper (Ye et al., 2024). We provide a sample query and rule-based CoT reasoning as an example below.

Query:
The number of each Grab & Go Store's Canned Peaches equals the sum of each QuickMart's Corned Beef, each Colonial Quarter's Product and each Colonial Quarter's QuickMart.
The number of each QuickMart's Canned Peaches equals 0.
The number of each QuickMart's Corned Beef equals 4 more than each QuickMart's Canned Peaches.
The number of each Colonial Quarter's QuickMart equals 6 more than the difference of each QuickMart's Corned Beef and each QuickMart's Product.
The number of each Vintage Architecture District's Grab & Go Store equals the sum of each QuickMart's Canned Peaches and each Colonial Quarter's Product.
The number of each Grab & Go Store's Corned Beef equals 0 times each QuickMart's Corned Beef.
The number of each Vintage Architecture District's QuickMart equals the sum of each Colonial Quarter's QuickMart and each Colonial Quarter's Product.
How many Product does each Vintage Architecture District have?

Rule-based CoT:
<think>
Let's compute the answer step by step. - According to the information given, the number of each QuickMart's Canned Peaches is 0. Let's call it D. So D = 0.
- Now, we can find the number of each QuickMart's Corned Beef. Let's call it b. Then b = 4 + D = 4 + 0 = 4.
- We can then calculate the number of each QuickMart's Product. Let's call it Y. Then Y = b + D = 4 + 0 = 4.
- We can then calculate the number of each Colonial Quarter's QuickMart. Let's call it d. Then d = 6 + (b - Y) = 6 + (4 - 4) = 6.
- Then, let's denote the number of each Colonial Quarter's Product as V. Then V = d * Y = 6 * 4 = 24.
- Then, let's denote the number of each Vintage Architecture District's Grab & Go Store as v. Then v = D + V = 0 + 24 = 24.
- Now, we can find the number of each Grab & Go Store's Canned Peaches. Let it be c. Then c = b + V + d = 4 + 24 + 6 = 34.
- We can then calculate the number of each Grab & Go Store's Corned Beef. Let it be l. Then l = 0 * b = 0 * 4 = 0.
- Now, we can find the number of each Grab & Go Store's Product. Let's denote it as r. Then r = c + l = 34 + 0 = 34.
- We can then calculate the number of each Vintage Architecture District's QuickMart. Let's call it i. Then i = d + V = 6 + 24 = 30.
- We can then calculate the number of each Vintage Architecture District's Product. Let's call it x. Then x = v * r + i * Y = 24 * 34 + 30 * 4 = 936.
Thus, the answer is 936.
</think>
<answer>
The final answer is $\boxed{936}$.
</answer>

Difficulty Control. Both KK and iGSM offer controllable difficulty. In iGSM, difficulty is determined by the *number of operations* required to reach the correct answer in the ground-truth rule-based reasoning trace. In KK, difficulty is controlled by the *total number of characters* (knights and knaves) involved in the puzzle. For KK, the SFT dataset includes puzzles with 4–8 characters, and the RL dataset spans 7–11. The in-distribution and out-of-distribution (OOD) evaluations use puzzles with 7–11 and 12–13 characters, respectively. For the iGSM-medium task, the SFT dataset contains problems with 15–20 operations, and the RL dataset contains 15–20. The in-distribution and OOD evaluation sets use 15–20 and 21–25 operations, respectively. For the iGSM-hard task, the SFT dataset again includes 15–20 operations, while the RL dataset contains 25–30. The in-distribution and OOD evaluation sets use 25–30 and 31–35 operations, respectively.

A.2 Training Details

The demonstration dataset size for the SFT stage is set to 8,000, and the RL dataset contains 10,000 examples. For RL, we adopt the DAPO algorithm (Yu et al., 2025c) with a rule-based outcome reward based on final answer accuracy. All experiments are conducted using the Verl framework (Sheng et al., 2025).

SFT Training. We perform supervised fine-tuning (SFT) on datasets curated using both Tailor and baseline methods. The key hyperparameters for SFT in the iGSM and KK tasks are summarized in Table 1. For dataset construction, we use 1,000 seed queries from the iGSM benchmark and 500 from the KK dataset.

Table 1: Configurations of SFT training

Configurations	Value
training epochs	4
global batch size	32
learning rate	5×10^{-6}
learning rate scheduler	constant
padding	not removed

RL training. We adopt DAPO (Yu et al., 2025c) for the main experiments using the Verl training framework (Sheng et al., 2025). The key hyperparameters for DAPO are listed in Table 2. For the GRPO experiments shown in Figure 9, we also set the overlong buffer length to 3,072 tokens and the overlong penalty factor to 1.0 to mitigate CUDA out-of-memory issues during RL training.

RL reward verifiers. As described in Section 2, we use an outcome-based reward in the RL process. In both the KK and iGSM tasks, the final answer is extracted via string matching. For the KK experiments, the SFT demonstration template is as follows:

<think>
... Thinking process ...
</think>
<answer>
Thus, the final answer is boxed{ {name} is a {role}, ...}.
</answer>

Here, name refers to the character's full name, and role refers to either Knave or Knight. To verify the correctness of the final answer, we examine

Table 2: Key hyperparameters for RL (DAPO) training

Configurations	Value
training epochs	5
batch size	$128 \times 16 = 2048$
sequence parallel size	2 (actor), 1 (ref)
gradient clipping	1.0
entropy coefficient	0.0
learning rate	1×10^{-6}
weight decay	0.1
warmup steps	10
optimizer	Adam
responses per prompt	16
max response length	4000 tokens
temperature	1.0
top-p	1.0
top-k	-1
KL loss coefficient	0.0
clip ratio (low)	0.2
clip ratio (high)	0.28
remove padding	True
overlong penalty factor	1.0
overlong buffer length	3072 tokens
rollout backend	vllm (Kwon et al., 2023)

each name–role pair using string matching. A completion is rewarded with $r = 1$ only if all roles are correctly assigned. Otherwise, the trace is labeled with $r = 0$.

For the iGSM task, we also use string matching in the reward function. The SFT demonstration template is as follows:

<think>
... Thinking process ...
< /think>
<answer>
The final answer is $\boxed{\{answer\}}$.
< /answer>

where the answer is always an integer. We assign $r = 1$ if the answer matches the ground-truth value, and 0 otherwise.

SFT curation pipeline details. In the first step of our Tailor pipeline, where student models generate demonstrations and the teacher model analyzes the traces, we provide the teacher with three randomly selected incorrect traces and one randomly selected correct trace. The teacher is instructed to analyze the failure cases and identify the reasoning behind the correct trace. We prompt the teacher model to generate Chain-of-Thought (CoT) reasoning enclosed between special tokens, followed by the generation of reasoning primitives (i.e., prompts used to curate the SFT dataset in the next step). The prompt used in this process is provided in Appendix B.1. Additionally, we include examples of primitives in Appendix B.2.

B Prompts

B.1 Prompts for trace analysis and primitive synthesis.

You are a large language model. Follow the instructions below carefully. Your goal is to generate instruction prompts that guide student models to produce high-quality reasoning.
(1) Below is a correct demonstration generated by a student model for the given query. Analyze the reasoning process and identify which behaviors or strategies are particularly effective and beneficial for the model's reasoning.
Query (Correct Case): {correct query}
Correct CoT: {correct response}
(2) Below are three failure cases, each consisting of the original query, the incorrect response from the student model, and the expected ground-truth answer. Analyze why each response is incorrect by referencing specific steps or reasoning patterns that led to the failure. Based on this analysis, provide instructions on how to identify and revise the failed demonstration to arrive at the correct answer.
Failure Case 1:
Query: {fail1 query}
Response: {fail1 response}
Ground Truth: {fail1 gt}
Failure Case 2:
Query: {fail2 query}
Response: {fail2 response}
Ground Truth: {fail2 gt}
Failure Case 3:
Query: {fail3 query}
Response: {fail3 response}
Ground Truth: {fail3 gt}
(3) Based on the comparison between the correct and failed demonstrations, explain what kind of reasoning behavior

is essential for robust and correct student model performance. You should explicitly go through each failure case one by one using the ground-truth answer, identify the correct reasoning pattern, uncover any implicit assumptions present in the correct reasoning path that lead to the correct final answers, and analyze how to correct the incorrect reasoning chains. Then, new instruction prompts are designed to: (a) preserve the reasoning pattern exhibited in the correct demonstration; (b) incorporate the reasoning strategies and implicit assumptions necessary to reach the correct answers in the failure cases; and (c) guide the language model to self-identify and self-correct when similar failure patterns arise, steering it toward the correct reasoning path.
Please perform the entire thinking process described above within the <prompt_think>...< /prompt_think> tag.
Please output the modified instruction prompt clearly inside <generated_prompt>...< /generated_prompt> tags.
Please do not include any demonstration example inside <generated_prompt>...< /generated_prompt>.
You should be aware that every query has a valid answer. So the generated prompt must encourage the language model to conclude a valid answer.
Do not include unrelated words such as < |im_start| > inside <generated_prompt>...< /generated_prompt>, just instructions for solving the problem.

B.2 Examples of synthetic reasoning primitive

Example primitives for the iGSM Dataset. The generated primitive examples for solving this math dataset are shown in the following boxes with simplification for display. Example 1 aims to prevent errors where the model jumps to answers without properly resolving dependencies or dropping intermediate relationships. It addresses these issues by requiring the model to explicitly construct the dependency graph, clearly identify the target value to solve, and conduct a more detailed examination, including recomputing key steps even when no errors are detected. Example 2 focuses on preventing misinterpretation of the given conditions or overlooking parts of the problem description. For instance, it helps the model avoid skipping constraints or variables that may appear redundant. The most distinctive parts in Examples 1 and 2 are highlighted in red and blue, respectively. All generated primitives are combined with format instructions to ensure the teacher model generations comply with the required output format.

iGSM Example primitive (prompt) 1:
Problem-Solving Instructions for Robust Reasoning
1. Define Variables and Equations:
- List every entity and attribute mentioned in the query.
- Assign a unique variable to each quantity.
- Translate all given statements into equations using these variables.
2. Prioritize Independent Variables:
- Solve variables with direct numerical assignments first (e.g., constants like ‘X = 5‘).
- Substitute these values into the dependent equations immediately.
3. Build Dependency Graphs:
- For each unsolved variable, list all equations where it appears.
- Identify the simplest equation to resolve next (e.g., least dependencies).
4. Solve Step-by-Step:
- Proceed incrementally, substituting known values into dependent equations.
5. Validate Intermediate Results:
- Recompute critical steps to verify consistency.
- Flag contradictions for re-evaluation.
6. Target-Focused Resolution:
- Clearly state the goal variable.
- Back-substitute from the goal to ensure all required variables are resolved.
7. Final Answer:
- Conclude with a boxed numerical answer (‘boxed{N}‘) supported by the validated reasoning chain.
Note: All variables are solvable.

iGSM Example primitive (prompt) 2:
Instructions for Solving the Problem:
1. Define All Variables Explicitly:
- Assign variables to each entity and relationship mentioned, including composite terms (e.g., "Enclosure" = sum of sub-components).
- Label all variables clearly (e.g., (X_Store) for "X at Store").
2. Translate Statements into Equations:
- Convert every given statement into a mathematical equation, even if it seems redundant.
- Preserve all operations (sums, differences, multipliers) exactly as stated.
3. Substitute Known Values Early:
- Substitute fixed values (e.g., "equals 9") immediately to simplify equations.
- Do not assume unconstrained variables are zero unless explicitly stated.
4. Explore Indirect Relationships:
- Track how variables influence others indirectly (e.g., if (A = B + C) and (B) depends on (D), express (A) in terms of (D)).
- If a variable's value is unresolved, check if it can be expressed in terms of other known variables.
5. Self-Correct for Consistency:
- If equations lead to contradictions (e.g., (0 = 1)), revisit earlier steps to identify missed relationships or incorrect assumptions.
- Verify that all given statements are fully utilized in the solution.
6. Conclude Rigorously:
- Ensure every step logically follows from the previous ones.
- If ambiguity remains, exhaustively test plausible interpretations to understand the problem and conditions (e.g., "Enclosure" as a sum) to arrive at a valid answer.

C Related Works

LLM Reasoning. Reasoning models were first conceptualized in the OpenAI series models (Jaech et al., 2024), referring to LLMs that leverage test-time scaling by generating long chains of thought (CoTs) before final answers to improve output quality (Guo et al., 2025; Team et al., 2025b). The success of OpenAI-o1 (Jaech et al., 2024) and DeepSeek-R1 (Guo et al., 2025) demonstrated that large-scale RL can incentivize reasoning capabilities in LLM training. Considerable efforts have been devoted to developing RL algorithms, such as VinePPO (Feng et al., 2023), Reinforce++ (Hu, 2025), and DAPO (Yu et al., 2025c), to advance the frontier of reasoning models. RL-based training has also been adopted in a variety of domain-specific LLMs in addition to general-purpose models (Liu et al., 2025; Shen et al., 2025a; Chu et al., 2025; Nguyen et al., 2025). Furthermore, LLM agents apply RL to enhance reasoning in textual tool use and multi-step planning (Song et al., 2025; Chen et al., 2025b; Jin et al., 2025; Qian et al., 2025), mobile and web environments (Chen et al., 2025a; Qi et al., 2024), and code generation (Wei et al., 2025; Zeng et al., 2025a; Pan et al., 2024).

Data-Centric Methods for RL Training. Data-centric approaches focus on improving the quality of training data rather than modifying the training algorithms (Zhou et al., 2023; Li et al., 2025; Guha et al., 2025; Yu et al., 2025b; Liang et al., 2025). Several works (Hong et al., 2023; Yao et al., 2024; Lee et al., 2024b) aim to shift the behavior policy distribution to facilitate more effective RL training (Yu et al., 2025a). Within the warm-start RL pipeline, where SFT precedes RL tuning, recent studies have shown that SFT induces coarse-grained changes in the LLM's thinking pattern distribution (Fu et al., 2025), and that RL post-training tends to amplify patterns learned during SFT (Zhao et al., 2025). These findings highlight the critical role of SFT demonstrations (Yan et al., 2025; Ma et al., 2025), which serve as a "format teacher" (He et al., 2025) to guide policy rollouts and exploration during RL. Our method falls within the data-centric category, focusing specifically on curating the SFT dataset to better prepare LLMs for downstream RL training.

Reasoning Primitives. Primitives are often used to represent to capture trajectory features (Goyal et al., 2020; Peng et al., 2019). In the context of LLM reasoning, the internal thinking patterns manifested in model completions have been referred to as reasoning primitives (Li et al., 2025), behaviors (Zhao et al., 2025; Cen et al., 2025), or reasoning strategies (Qu et al., 2025). Prior works have identified specific primitives, such as reflection and backtracking, as correlates of effective test-time scaling and RL performance improvements (Yeo et al., 2025; Shen et al., 2025b; Kim

et al., 2025). Additionally, by comparing models that show large versus marginal gains during RL, Gandhi et al. (2025) identified four key primitives: verification, backtracking, backward chaining, and subgoaling, which are critical to RL success. In contrast to these studies, which focus on analyzing specific reasoning patterns, we explore how to automatically discover novel primitives and investigate how the diversity and quality of reasoning primitives affect the effectiveness of RL.

D Usage of LLMs

We used publicly available LLMs to assist with proofreading and grammar refinement of the paper draft. All technical content was verified by the authors. Our research also involves LLMs as a core component: we distill LLMs to curate datasets and also fine-tune LLMs. The authors take full responsibility for all research ideas, technical contributions, and conclusions.