From Implicit Exploration to Structured Reasoning: Leveraging Guideline and Refinement for LLMs

Jiaxiang Chen[1] **Zhuo Wang**[1,2] **Mingxi Zou**[1] **Zhucong Li**[1]
Zhijian Zhou[1,2] **Song Wang**[4] **Zenglin Xu**[1,3]*
[1]Fudan University [2]Shanghai Innovation Institute(SII)
[3]Shanghai Academy of AI for Science (SAIS)
[4]Zoom Video Communications
jiaxiangchen23@m.fudan.edu.cn

Abstract

Large language models (LLMs) have advanced general-purpose reasoning, showing strong performance across diverse tasks. However, existing methods often rely on implicit exploration, where the model follows stochastic and unguided reasoning paths—like walking without a map. This leads to unstable reasoning paths, lack of error correction, and limited learning from past experience. To address these issues, we propose a framework that shifts from implicit exploration to structured reasoning through *guideline* and *refinement*. First, we extract structured reasoning patterns from successful trajectories and reflective signals from failures. During inference, the model follows these guidelines step-by-step, with refinement applied after each step to correct errors and stabilize the reasoning process. Experiments on BBH and four additional benchmarks (GSM8K, MATH-500, MBPP, HumanEval) show that our method consistently outperforms strong baselines across diverse reasoning tasks. Structured reasoning with stepwise execution and refinement improves stability and generalization, while guidelines transfer well across domains and flexibly support cross-model collaboration, matching or surpassing supervised fine-tuning in effectiveness and scalability.

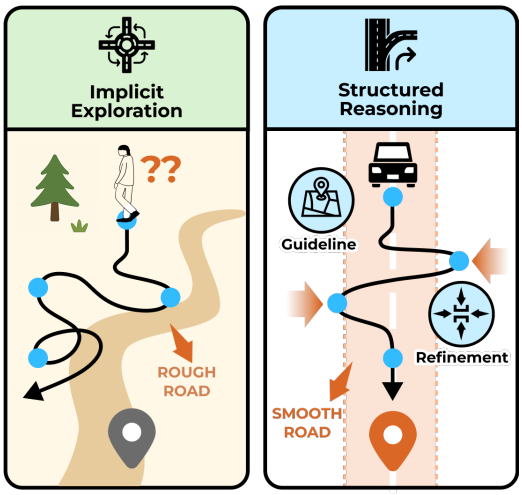

Figure 1: Comparison between **Implicit Exploration** and **Structured Reasoning**. **Left:** Implicit exploration is like walking on a rough path without a map—lacking clear direction, it often leads to unstable and error-prone reasoning. **Right:** Structured reasoning resembles driving with a roadmap: the *guideline* offers a global route, *refinement* helps correct deviations along the way, and the entire reasoning process remains smoother and more stable—like navigating a wide, well-marked road.

1 Introduction

Large language models (LLMs) (Achiam et al., 2023; OpenAI, 2024; Meta AI, 2024) have demonstrated strong generalization across diverse domains, including mathematics (Imani et al., 2023), logical reasoning (Pan et al., 2023), language understanding (Nam et al., 2024; An et al., 2024), and specialized applications such as finance (Yu et al., 2025; Chen et al., 2025). While in-context learning (ICL) enables flexible adaptation, LLMs continue to struggle with complex multi-step reasoning tasks—including planning, symbolic manipulation, and structured decision-making—which are essential for many real-world applications. Overcoming these limitations is critical for improving both the theoretical understanding and practical reliability of LLM-based systems.

Recent work has made progress toward this goal. Many researchers have explored advanced reasoning paradigms built on Chain-of-Thought (CoT)(Wei et al., 2022; Brown et al., 2020; Kojima et al., 2022; Wang et al., 2022) prompting. ReAct(Yao et al., 2023b) introduces a "thought–action–observation" cycle that enables interaction with the environment and self-evaluation. Tree-of-Thought (ToT)(Yao et al., 2023a) represents reasoning as a search over tree-structured trajectories, encouraging exploration of multiple

* Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 3672–3684
November 4-9, 2025

possibilities. Beats (Sun et al., 2024) and FoT(Bi et al., 2024) use MCTS(Chaslot et al., 2008) and voting to improve robustness. Notably, few-shot CoT and FoT attempt to leverage prior examples to improve reasoning. Although these approaches represent progress toward more structured reasoning, they still operate under an implicit reasoning paradigm: they leverage past examples in limited ways, lack global guidance, and offer no principled mechanism for error correction.

Nevertheless, implicit reasoning methods face three persistent challenges in complex multi-step tasks. First, models often struggle to learn from prior experience and extract reusable strategies—reasoning is executed from scratch without leveraging past successes. Second, reasoning trajectories are frequently unstable. In the absence of a well-defined global structure, models tend to deviate from valid reasoning paths, leading to the compounding of early-stage errors. Third, existing methods lack mechanisms for error recovery; once a reasoning path is taken, there is typically no reflection or correction. As illustrated in Figure 1, these limitations underscore the need to move from implicit exploration toward structured reasoning guided by learned strategies and iterative refinement.

Therefore, we propose a structured reasoning framework based on guideline extraction and stepwise refinement to address three key challenges in complex reasoning: unstable reasoning paths, lack of error correction, and limited use of learned experience. The framework consists of two stages. First, we extract structured reasoning patterns and key decision points from successful trajectories to form generalizable **guidelines**. In parallel, we analyze failed cases to identify typical error patterns, which serve as reflective signals. Second, during inference, the model follows the extracted guideline step by step. After each step, a refinement module evaluates the intermediate output and applies targeted corrections. This procedure introduces global planning, enables error recovery, and incorporates experiential learning.

We evaluate our approach on complex reasoning tasks from the Big-Bench Hard (BBH) benchmark (Suzgun et al., 2023) across multiple models. Our framework consistently outperforms strong baselines—including CoT, ReAct, ToT, Beats and FoT—achieving notable gains in both accuracy and stability. Further analysis reveals that step-wise execution enhances reasoning coherence through explicit decomposition, refinement enables real-time error correction, and experience-based learning produces effective instructions that outperform implicit self-planning. We also investigate model collaboration in the refinement stage, providing insights into cross-model interactions and division of labor. Additionally, inspired by prior work (Dai et al., 2023) that frames ICL as implicit optimization, we compare our method to supervised fine-tuning (SFT) using the same model (LLaMA-3.3-70B), and observe better generalization across tasks. These findings suggest that structured reasoning with guideline and refinement not only improves interpretability and reliability, but also holds promise for scalable deployment in more complex and practical real-world scenarios.

Our main contributions are:

- We introduce a structured reasoning framework that transitions from implicit exploration to explicit process modeling through *guideline extraction* and *stepwise refinement*.
- We propose and validate three mechanisms for structured reasoning: stepwise execution improves stability, refinement enables error correction, and experience-based learning produces reusable strategies.
- Our method consistently outperforms strong baselines on BBH tasks across model scales. We further investigate inter-model collaboration during refinement and demonstrate that our structured approach surpasses supervised fine-tuning (SFT) on LLaMA-3.3-70B, offering a scalable and interpretable alternative for complex reasoning.

2 Related Work

2.1 In-Context Learning

In-Context Learning (ICL) (Zhao et al., 2023; Huang and Chang, 2022) allows LLMs to perform tasks by conditioning on input examples without updating parameters (Kojima et al., 2022; Brown et al., 2020; Dong et al., 2024; Zhang et al., 2025). Auto-CoT (Zhang et al.) enriches prompts by generating reasoning chains, while Many-shot (Agarwal et al., 2024) and From Few to Many (Wan et al., 2025) improve performance by selecting better exemplars, which often via optimization over example pools. Despite better exemplars, prior ICL methods fail to abstract reasoning structure—our

approach learns structured guidelines to guide stable and adaptive reasoning.

2.2 Chain-of-Thought Reasoning

Chain-of-Thought (CoT) (Wei et al., 2022; Huang and Chang, 2022; Hao et al., 2023) enhances interpretability via step-wise reasoning, while ReAct (Yao et al., 2023b) integrates reasoning with actions. Tree-of-Thought (ToT) and its variants (Yao et al., 2023a; Besta et al., 2024; Bi et al., 2024) structure inference as multi-path search. Recent methods like Beats (Sun et al., 2024), FoT (Bi et al., 2024), and HiAR-ICL (Wu et al., 2024) employ MCTS to improve exploration and consistency. However, they often overlook fine-grained step correction. In contrast, our method extracts structured plans and applies step-wise refinement for more stable and efficient reasoning.

3 Method

We propose a structured reasoning framework that transitions from implicit exploration to explicit, guideline-driven reasoning. Our method is built on two complementary components:

Guideline Learning: Automatically extracts stable reasoning patterns from correct examples and summarizes reflective signals from failure cases to form structured, step-wise guidelines.

Guided Execution with Refinement: Leverages learned guidelines to conduct reasoning step-by-step during inference, with refinement applied to each intermediate result for enhanced robustness.

3.1 Guideline Learning

To improve reasoning reliability, we introduce an automatic guideline learning module that distills structured reasoning steps from both correct and incorrect model trajectories.

Given an input x, the model first generates an initial reasoning process $R = (r_1, r_2, \ldots, r_T, \hat{y})$, where r_t corresponds to the t-th step in the reasoning process. The final output $\hat{y}$ is then compared with the ground-truth label y^*:

- **If** $\hat{y} = y^*$, we extract key reasoning steps and summarize stable reasoning patterns.

- **If** $\hat{y} \neq y^*$, we analyze the trajectory to identify common reasoning failures and potential improvements.

We define two functions: f_{ext} extracts effective reasoning patterns from (x, R, y^*), producing guideline candidates $\mathcal{G}$. f_{ref} analyzes failures in (x, R, y^*) to generate mistake-aware reflections $\mathcal{M}$:

$$\mathcal{G} = f_{\text{ext}}(x, R, y^*)$$

$$\mathcal{M} = f_{\text{ref}}(x, R, y^*)$$

After processing all samples in the train-set of dataset $\mathcal{D}$, we aggregate the learned guidelines into a final structured guideline steps $\mathcal{G}_T$,where f_{agg} induces patterns across all examples to form $\mathcal{G}_T = (\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_T)$, among them T is the length of reasoning steps.

$$\mathcal{G}_T = f_{\text{agg}}(\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \ldots, \mathcal{G}^{(N)})$$

Here, $\mathcal{G}_T = (\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_T)$ represents a step-wise reasoning guideline synthesized across all examples.

Algorithm 1 Guideline Learning

Require: Training set $\mathcal{D}$ with samples (x, y^*)
Initialize reasoning guideline buffer
$\mathcal{G}_{\text{buffer}} \leftarrow \emptyset$
for each sample $(x, y^*) \in \mathcal{D}$ **do**
    Generate initial reasoning path
    $R = (r_1, r_2, \ldots, r_T, \hat{y})$
    if $\hat{y} = y^*$ **then**
        Extract reasoning steps
        $\mathcal{G} = f_{\text{ext}}(x, R, y^*)$
    else
        Reflect on errors and prevent mistakes
        $\mathcal{M} = f_{\text{ref}}(x, R, y^*)$
    end if
    Store:
    $\mathcal{G}_{\text{buffer}} \leftarrow \mathcal{G}_{\text{buffer}} \cup \mathcal{G}$ *or* $\mathcal{M}$
end for
Aggregate final guideline: $\mathcal{G}_T = f_{\text{agg}}(\mathcal{G}_{\text{buffer}})$
return Learned guideline steps $\mathcal{G}_T$

3.2 Guided Execution with Refinement

Once guidelines $\mathcal{G}_T$ are learned, we use them to guide step-wise inference and perform dynamic refinements to ensure reasoning stability.

For an input x and learned guideline set $\mathcal{G}_T$, the model executes reasoning step-by-step:

$$r_t = f_{\text{execute}}(x, \mathcal{G}_t)$$

where f_{execute} generates the current reasoning step r_t based on input x and the structured guideline $\mathcal{G}_T$.

After executing each step r_t, we inspect the result for common errors captured in the guideline $\mathcal{G}_t$.

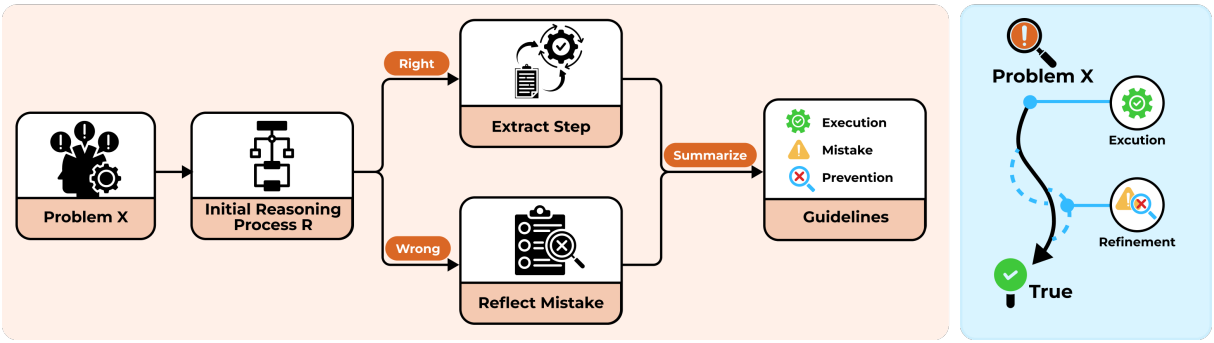

Figure 2: Overview of our framework. **Left:** The Guideline Learning module extracts reasoning steps from successful cases and identifies common mistakes from failed ones, summarizing them into generalizable *guidelines*. **Right:** During inference, the model follows the learned guidelines step by step, with continuous *refinement* to correct errors and improve reasoning stability.

When an issue is found, we apply the associated prevention strategy to produce a refined step.

We define an function f_{refine} that refines the reasoning step:

$$r_t^* = f_{\text{refine}}(x, r_t, \mathcal{G}_t)$$

where r_t^* is the refined step that improves reasoning stability.

After all reasoning steps are executed and refined, the final output is derived as,where $R = [r_1, r_2, ..., r_t]$ is the final refined reasoning steps:

$$\hat{y} = f_{\text{final}}(R)$$

where f_{final} integrates the reasoning sequence R to extract the final answer $\hat{y}$.

Algorithm 2 Guided Execution with Refinement

Require: Input x, learned guideline steps $\mathcal{G}_T$

Initialize reasoning path $R \leftarrow \emptyset$
for each step t **do**
  Execute current step
  $r_t = \text{execute}(x, R, \mathcal{G}_t)$
  Append to reasoning path
  $R \leftarrow R \cup \{r_t\}$
  Refine step result:
  $r_t \leftarrow f_{\text{refine}}(x, r_t, \mathcal{G}_t)$
end for
Extract final conclusion $y = f_{\text{final}}(R)$
return final output y

4 Experiments

4.1 Experimental Setup

Datasets We primarily conduct experiments on eight diverse subsets from the BBH benchmark: Causal Judgement, Formal Fallacies, Geometric Shapes, Hyperbaton, Logical Deduction_7, Navigate, Salient Translation Error Detection, and Multi-step Arithmetic. For clarity, we group these tasks into three categories and use their abbreviations throughout the paper. Following a consistent protocol, we randomly select 25% of each dataset for training (guideline extraction) and use the remaining 75% for evaluation, with all reported results based on the held-out test sets.

Mathematical Reasoning: Geometric Shapes (GS), Multi-step Arithmetic (MA), Navigate (NA)

Logical Reasoning: Causal Judgement (CJ), Formal Fallacies (FF), Logical Deduction_7 (LD)

Content Understanding: Hyperbaton (HY), Salient Translation Error Detection (ST)

To further assess generality beyond BBH, we additionally evaluate on GSM8K, MATH-500, MBPP, and HumanEval. For all these datasets, we follow the same sampling strategy: 25% of the data is used for training guideline extraction, and the remaining 75% for evaluation.

Models We evaluate on both proprietary and open-source language models. GPT-4o and GPT-4o-mini serve as our primary closed-source models, while LLaMA-3.1-8B-Instruct and Qwen3-8B represent strong open-weight baselines. To study the transferability of structured reasoning, especially in distilling long CoT chains, we also include LLaMA-3.3-70B-Instruct and DeepSeek-R1-Distill-LLaMA-70B. For brevity, we refer to GPT-4o, GPT-4o-mini, LLaMA-3.1-8B-Instruct, and Qwen3-8B as 4o, mini, llama, and qwen, respectively.

Baselines We compare against two categories of baselines:

CoT-based Methods. CoT (Wei et al., 2022) prompts models to solve problems step by step. Few-shot CoT further provides annotated examples to guide human-like reasoning through demonstration.

Reasoning Frameworks. ReAct (Yao et al., 2023b) combines reasoning with actions and observations for interactive problem solving. Tree-of-Thought (ToT) (Yao et al., 2023a) explores multiple solution paths via tree search ($n = 3$). Beats (Sun et al., 2024) uses MCTS with majority voting to enhance diversity and robustness ($d_{max} = 5$, $a_{max} = 3$). FoT (Bi et al., 2024) adopts MCTS within a multi-tree reasoning structure ($n = 3$, $iter_{max} = 2$).

Evaluation Metrics All models are evaluated using **accuracy**, where the model-generated answer is extracted from the `<answer>` tag. This ensures consistent answer alignment and allows for an objective comparison across different models and reasoning approaches.

4.2 Comparison with Chain-of-Thought

As illustrated in Table 1,Table 3 and Figure 3, our method consistently surpasses both CoT and Few-shot CoT across eight reasoning tasks, spanning mathematical, logical, and content understanding categories. The gains are especially pronounced on multi-step and long-horizon problems, and remain robust across model scales from LLaMA-3.1-8B to GPT-4o.

These findings highlight the limitations of conventional CoT approaches, which rely on implicit reasoning and static demonstrations. Even with few-shot examples, models often fail to derive transferable strategies or to recover from accumulated errors, resulting in unstable reasoning. In contrast, our framework integrates structured guidelines to direct each step and a refinement mechanism for adaptive correction, producing more stable and accurate performance.

4.3 Comparison with Reasoning Frameworks

As shown in Table 2 and Figure 3, our method consistently outperforms ReAct, Tree-of-Thought (ToT), Beats, and FoT across three model scales: LLaMA-3.1-8B, GPT-4o-mini, and GPT-4o. The gains hold across all BBH task categories—mathematical, logical, and content understanding—demonstrating the robustness of structured guideline-based reasoning. Furthermore, Table 3 shows that these improvements generalize beyond BBH, with the mini model achieving strong results on mathematical (GSM8K, MATH-500) and code generation (MBPP, HumanEval) benchmarks.

These results suggest that guideline-driven reasoning provides a more stable and efficient alternative to existing frameworks. While ReAct and ToT rely on trial-and-error exploration or heuristic tree search, and Beats on costly MCTS sampling, FoT improves performance through multi-tree exploration with limited reuse of thought processes. In contrast, our framework unifies global planning via learned guidelines with local adaptability through refinement, enabling targeted optimization and dynamic correction—particularly effective for multi-step reasoning tasks.

4.4 Detailed Analysis

We conduct a comprehensive analysis of the three core mechanisms in our framework: stepwise execution, refinement, and guideline-based learning. These mechanisms are evaluated for their impact on performance, stability, and generalizability in reasoning tasks. We present ablation studies and inter-collaboration patterns under different model configurations.

Table 4 summarizes the performance under different configurations. Each setting toggles whether guideline is learned from past experience, stepwise execution, and refinement are enabled. Results are reported for GPT-4o, GPT-4o-Mini, LLaMA-3.1-8B and Qwen3-8B.

4.4.1 Stepwise Execution Mechanism

According to Table 4, for both the 4o and mini models, stepwise execution consistently outperforms single-step reasoning, regardless of whether the refinement mechanism is enabled. This highlights the benefit of structuring the reasoning process into sequential steps to enhance stability and robustness.

However, an interesting deviation is observed with the llama model, which performs worse under full refinement compared to simpler settings. In contrast, the similarly sized qwen model does not exhibit this degradation, suggesting that the effect is not purely due to model scale but rather tied to differences in reasoning capability. We further investigate this phenomenon in Section 4.4.3 through inter-model execution and refinement experiments.

To better understand the role of execution granularity, we examine the impact of step count using both the 4o and mini models (note: in Table 4, the

Table 1: Performance comparison between CoT-based methods and our approach across eight reasoning tasks. The datasets are grouped into three categories: Mathematical Reasoning (GS, MA, NA), Logical Reasoning (CJ, FF, LD), and Content Understanding (HY, ST).

Model	Method	Mathematical Reasoning			Logical Reasoning			Content Understanding		Avg
		GS	MA	NA	CJ	FF	LD	HY	ST	
4o	CoT	0.631	0.968	0.979	0.621	0.476	0.620	0.882	0.695	0.734
	Few-shot CoT	0.599	0.978	0.973	0.593	0.636	0.813	0.989	0.787	0.789
	Ours	0.711	0.963	0.973	0.671	0.770	0.925	0.995	0.888	**0.862**
mini	CoT	0.455	0.957	0.968	0.636	0.743	0.508	0.775	0.583	0.703
	Few-shot CoT	0.551	0.930	0.963	0.636	0.733	0.604	0.957	0.631	0.751
	Ours	0.658	0.936	0.952	0.657	0.679	0.711	0.995	0.813	**0.800**
llama	CoT	0.267	0.428	0.497	0.493	0.251	0.272	0.529	0.639	0.422
	Few-shot CoT	0.679	0.455	0.674	0.507	0.369	0.203	0.513	0.567	0.496
	Ours	0.583	0.401	0.529	0.621	0.444	0.572	0.807	0.717	**0.584**

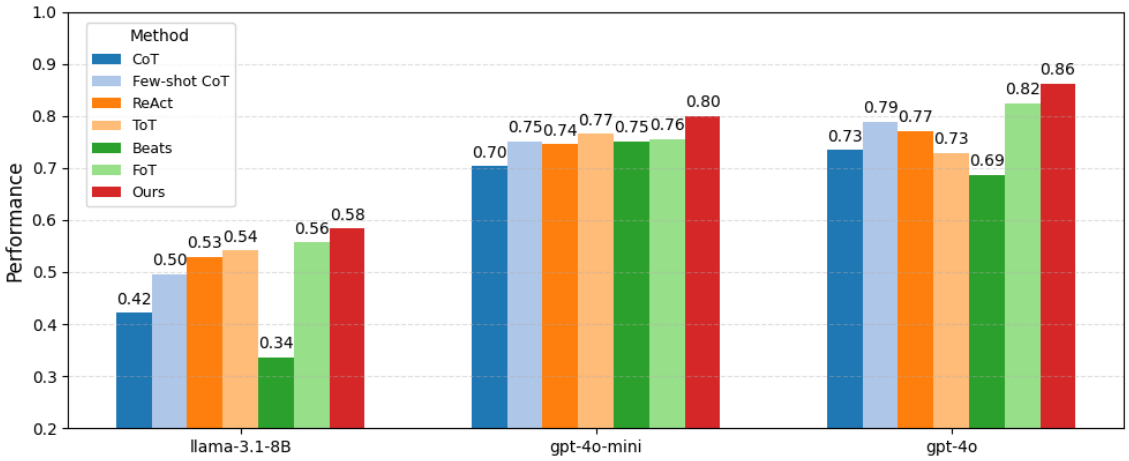

Figure 3: **Overall performance comparison across six reasoning methods under different model scales**. The methods include: **CoT-based** approaches (CoT, Few-shot CoT), which rely on implicit pattern imitation; and **reasoning frameworks** (ReAct, ToT, Beats,FoT), which introduce dynamic interaction, search, or voting-based selection. Our structured approach (Ours) consistently achieves superior performance, demonstrating the effectiveness of guideline-based execution and stepwise refinement in complex reasoning tasks.

number of steps is not strictly fixed but varies by task, typically ranging from 6 to 10). As shown in Figure 4, increasing the number of reasoning steps from one to five yields substantial performance gains, suggesting that stepwise decomposition enhances stability and decision quality. Beyond five steps, however, the improvement plateaus, indicating diminishing returns.

4.4.2 Refinement Mechanism

As presented in Table 4, applying refinement improves performance across both stepwise and single-step execution, indicating its general effectiveness in enhancing inference quality. While this pattern holds for 4o, Mini, and qwen, LLaMA exhibits a slight performance drop when refinement is applied, suggesting that the issue is not simply due to model size but rather reflects differences in reasoning robustness.

To further examine this effect, we vary the number of refinement rounds, as shown in Figure 5. For 4o, performance peaks after a single refinement, implying that one iteration of self-correction is sufficient for strong models to stabilize their reasoning. In contrast, Mini achieves optimal performance with two refinement rounds, indicating that moderately sized models benefit from additional correction to offset weaker initial outputs. However, further refinement beyond the optimal point introduces noise or over-adjustment, leading

Table 2: Performance comparison between our approach and other reasoning frameworks across eight reasoning tasks, grouped by category.

Model	Method	Mathematical Reasoning			Logical Reasoning			Content Understanding		Avg
		GS	MA	NA	CJ	FF	LD	HY	ST	
4o	ReAct	0.545	0.620	0.995	0.619	0.807	0.925	0.872	0.786	0.771
	ToT	0.419	0.663	0.968	0.669	0.775	0.882	0.674	0.786	0.729
	Beats	0.545	0.989	0.984	0.633	0.283	0.684	0.850	0.524	0.687
	FoT	0.738	0.973	0.984	0.679	0.695	0.834	0.947	0.743	0.824
	Ours	0.711	0.963	0.973	0.671	0.770	0.925	0.995	0.888	**0.862**
mini	ReAct	0.481	0.952	0.952	0.669	0.690	0.802	0.775	0.636	0.745
	ToT	0.561	0.941	0.979	0.657	0.738	0.706	0.882	0.663	0.766
	Beats	0.545	0.947	0.979	0.679	0.551	0.727	0.963	0.620	0.751
	FoT	0.486	0.947	0.979	0.640	0.775	0.733	0.845	0.636	0.755
	Ours	0.658	0.936	0.952	0.657	0.679	0.711	0.995	0.813	**0.800**
llama	ReAct	0.417	0.465	0.454	0.471	0.524	0.460	0.780	0.663	0.529
	ToT	0.476	0.439	0.561	0.450	0.588	0.449	0.733	0.642	0.542
	Beats	0.209	0.267	0.652	0.471	0.241	0.241	0.310	0.299	0.336
	FoT	0.380	0.444	0.620	0.593	0.545	0.529	0.706	0.636	0.557
	Ours	0.583	0.401	0.529	0.621	0.444	0.572	0.807	0.717	**0.584**

Table 3: Performance on math (GSM8K, MATH-500) and code (MBPP, HumanEval) benchmarks using the **mini** model.

Method	Math		Code	
	GSM8K	MATH-500	MBPP	HumanEval
CoT	0.872	0.578	0.663	0.886
ReAct	0.782	0.580	0.690	0.920
ToT	0.901	0.652	0.673	0.900
Beats	0.906	0.659	0.700	0.935
FoT	0.904	0.658	0.691	0.931
Ours	**0.921**	**0.676**	**0.730**	**0.938**

Table 4: Performance comparison across different configurations. ✓ indicates the component is enabled, ✗ means disabled.

Learn	Step	Refine	4o	mini	llama	qwen
✓	✓	✓	**0.862**	**0.800**	0.584	**0.820**
✓	✓	✗	0.802	0.785	0.635	0.803
✓	✗	✓	0.801	0.762	0.646	0.787
✓	✗	✗	0.799	0.760	**0.650**	0.770
✗	✓	✓	0.809	0.634	0.499	–

to performance degradation in both models. These findings highlight the importance of calibrating refinement depth based on model capacity.

4.4.3 Inter-Model Collaboration

Table 4 shows both 4o and mini benefit from the addition of stepwise execution and refinement mechanisms, exhibiting consistent performance improvements. In contrast, the performance of llama degrades when either mechanism is introduced. This suggests that smaller models may suffer from error accumulation and lack sufficient capacity for effective refinement.

To overcome this limitation, we explore *inter-model collaboration* by pairing models as executor and refiner. As shown in Figure 6, stronger refiners like GPT-4o consistently improve performance, while weaker refiners (e.g., LLaMA or mini) may degrade results, underscoring the need for complementary model roles.

4.4.4 Supervision and Transferability

A key question is whether our method requires perfectly task-aligned supervision. We evaluate guideline transferability in two settings:

(A) *In-domain transfer*: each test task uses guidelines from another task in the same domain. – MA (Math) from GS – CJ (Logic) from FF – HY (Content) from ST

(B) *Cross-domain transfer*: guidelines are transferred across domains, with GS (Math), FF (Logic), and ST (Content) serving as representative sources.

Results are shown in Table 5.

In-domain guidelines often perform on par with or better than task-specific ones, while cross-

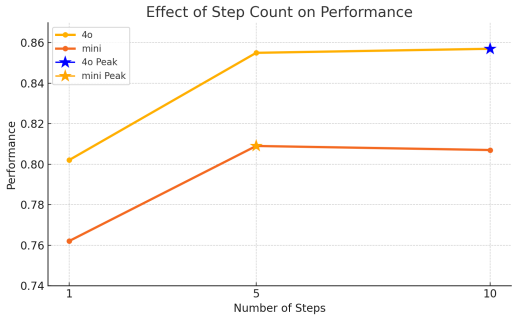

Figure 4: Effect of step count on performance

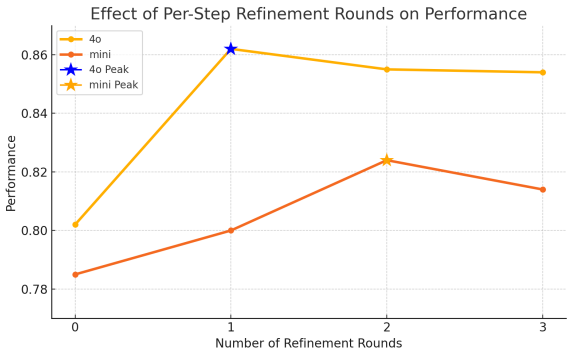

Figure 5: Effect of Per-Step Refinement Rounds on Performance

domain transfer also yields competitive results. These findings indicate that the method is robust to weaker supervision and moderate domain shifts.

4.4.5 Learning vs. Self-Planning

We compare models guided by learned guidelines with those that rely solely on implicit self-planning. In the self-plan setting, the model generates a step plan before execution, but lacks prior execution experience or task-specific guidance for avoiding common errors.

As depicted in the Figure 7, structured reasoning driven by learned experience consistently outperforms such unguided planning. Even for large models like 4o, learning-based execution yields noticeable gains, while the gap becomes more pronounced for smaller models like mini and llama.

4.4.6 SFT vs. Structured Reasoning

Inspired by prior works (Dai et al., 2023),we investigate whether structured reasoning can be effectively distilled from a stronger model and transferred to others. We compare two approaches: **(1) Guideline-Assisted Reasoning**, which extracts stepwise reasoning traces from R1's Chain-of-Thought (CoT) and applies them to other models; and **(2) SFT-Based Distillation**, which uses the

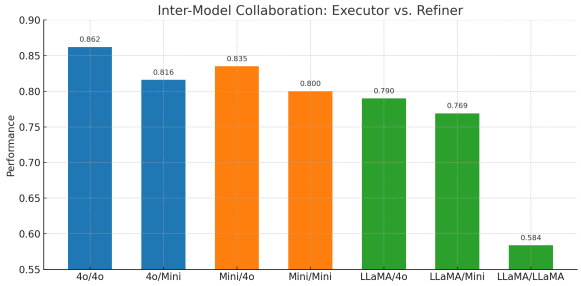

Figure 6: Performance of inter-model collaboration under different executor/refiner pairings. Models are grouped by the executor (4o, mini, llama). Within each group, using a stronger refiner generally improves performance.

Table 5: Guideline transferability under (A) in-domain and (B) cross-domain settings (accuracy). Domain labels: Math = Mathematical, Logic = Logical, Content = Content Understanding.

(A) In-domain transfer

Task	Task-Specific	In-Domain
MA (Math)	0.936	0.957
CJ (Logic)	0.657	0.620
HY (Content)	0.995	0.970

(B) Cross-domain transfer

Task	GS (Math)	FF (Logic)	ST (Content)
GS	0.658	0.513	0.668
FF	0.599	0.679	0.535
ST	0.743	0.786	0.813

distilled version of R1 (DeepSeek-R1-Distill-llama-3.3-70B) with standard CoT prompting to assess whether reasoning quality is preserved after fine-tuning.

As shown by Figure 8, both approaches are evaluated on the same base model, llama-3.3-70B. Our guided reasoning framework consistently outperforms the SFT-distilled variant, despite requiring no additional training. This highlights that structured reasoning can be effectively induced through lightweight external guidance, providing a more interpretable and flexible alternative to implicit learning via supervised fine-tuning.

4.5 Case Study:Geometry Shapes

To qualitatively evaluate the model's structured reasoning ability, we conduct a case study on the `geometric_shapes` task. The guideline extracted from training examples is shown in Appendix Figure 9, detailing step-wise instructions, common mistakes, and prevention strategies. Appendix Figure 10 demonstrates how GPT-4o applies this guide-

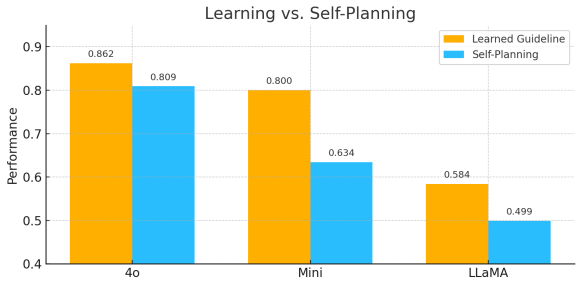

Figure 7: Guideline-based reasoning consistently outperforms self-planning.

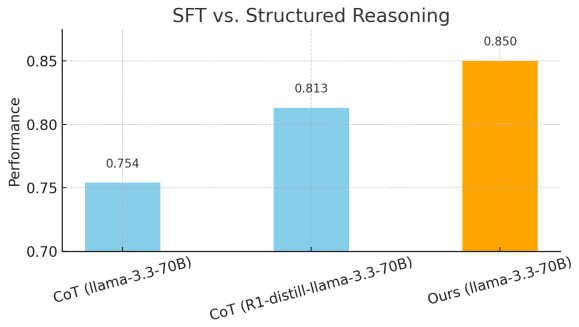

Figure 8: **SFT vs. Structured reasoning** Performance comparison between CoT prompting on the original model, its SFT-distilled variant, and our guideline-based reasoning framework (all using llama-3.3-70B).

line to a specific SVG input. Notably, an initial error in path closure detection is corrected through refinement, while all other steps are executed correctly.

5 Conclusion

We propose a structured reasoning framework that transitions from implicit exploration to explicit process modeling via guideline extraction and stepwise refinement. This approach enhances reasoning stability, supports error correction, and enables experience-based generalization. Experiments on eight BBH tasks across multiple models show consistent improvements over strong baselines, including CoT, ReAct, ToT, Beats and FoT. Further analysis highlights the contributions of stepwise execution, iterative refinement, and learned guidance. We also explore inter-model collaboration in the refinement stage and show that our method performs competitively with, and in some cases outperforms, supervised fine-tuning (SFT) approaches that distill knowledge from larger models—demonstrating strong scalability and practical potential for complex reasoning tasks.

Limitations

Although we explore inter-model collaboration during refinement, the design space of combining models at different scales remains underexplored. Future work may investigate more diverse and adaptive model configurations to further enhance efficiency, robustness, and scalability in practical deployments.

Acknowledgments

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Rishabh Agarwal, Avi Singh, Lei Zhang, Bernd Bohnet, Luis Rosias, Stephanie Chan, Biao Zhang, Ankesh Anand, Zaheer Abbas, Azade Nova, and 1 others. 2024. Many-shot in-context learning. *Advances in Neural Information Processing Systems*, 37:76930–76966.

Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. 2024. Make your llm fully utilize the context. *Advances in Neural Information Processing Systems*, 37:62160–62188.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and 1 others. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690.

Zhenni Bi, Kai Han, Chuanjian Liu, Yehui Tang, and Yunhe Wang. 2024. Forest-of-thought: Scaling test-time compute for enhancing llm reasoning. *arXiv preprint arXiv:2412.09078*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Guillaume Chaslot, Sander Bakkes, Istvan Szita, and Pieter Spronck. 2008. Monte-carlo tree search: A new framework for game ai. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 4, pages 216–217.

Jiaxiang Chen, Mingxi Zou, Zhuo Wang, Qifan Wang, Dongning Sun, Chi Zhang, and Zenglin Xu. 2025. Finhear: Human expertise and adaptive risk-aware temporal reasoning for financial decision-making. *arXiv preprint arXiv:2506.09080*.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. 2023. Why can gpt learn in-context? language models secretly perform gradient descent as meta-optimizers. In *ACL (Findings)*.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, and 1 others. 2024. A survey on in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1107–1128.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8154–8173.

Jie Huang and Kevin Chen-Chuan Chang. 2022. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403*.

Shima Imani, Liang Du, and Harsh Shrivastava. 2023. Mathprompter: Mathematical reasoning using large language models. *arXiv preprint arXiv:2303.05398*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Meta AI. 2024. Introducing llama 3.1. https://ai.meta.com/blog/meta-llama-3-1/.

Daye Nam, Andrew Macvean, Vincent Hellendoorn, Bogdan Vasilescu, and Brad Myers. 2024. Using an llm to help with code understanding. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, pages 1–13.

OpenAI. 2024. Hello gpt-4o. https://openai.com/index/hello-gpt-4o/.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. 2023. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. *arXiv preprint arXiv:2305.12295*.

Linzhuang Sun, Hao Liang, Jingxuan Wei, Bihui Yu, Conghui He, Zenan Zhou, and Wentao Zhang. 2024. Beats: Optimizing llm mathematical capabilities with backverify and adaptive disambiguate based efficient tree search. *arXiv preprint arXiv:2409.17972*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, and 1 others. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *ACL (Findings)*.

Xingchen Wan, Han Zhou, Ruoxi Sun, Hootan Nakhost, Ke Jiang, and Sercan Ö Arık. 2025. From few to many: Self-improving many-shot reasoners through iterative optimization and generation. *arXiv preprint arXiv:2502.00330*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

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.

Jinyang Wu, Mingkuan Feng, Shuai Zhang, Feihu Che, Zengqi Wen, and Jianhua Tao. 2024. Beyond examples: High-level automated reasoning paradigm in in-context learning via mcts. *arXiv preprint arXiv:2411.18478*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Yangyang Yu, Haohang Li, Zhi Chen, Yuechen Jiang, Yang Li, Jordan W Suchow, Denghui Zhang, and Khaldoun Khashanah. 2025. Finmem: A performance-enhanced llm trading agent with layered memory and character design. *IEEE Transactions on Big Data*.

Beichen Zhang, Yuhong Liu, Xiaoyi Dong, Yuhang Zang, Pan Zhang, Haodong Duan, Yuhang Cao, Dahua Lin, and Jiaqi Wang. 2025. Booststep: Boosting mathematical capability of large language models via improved single-step reasoning. *arXiv preprint arXiv:2501.03226*.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, and 1 others. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2).

A Appendix

A.1 Case Study: Structured Reasoning in Geometric Shapes Task

All results in this case study are generated using GPT-4o. Figure 9 outlines the structured guideline learned for reasoning over geometric shapes based on SVG path representations. Figure 10 provides a step-wise case study where the model applies this guideline to identify a triangle, including correction of an intermediate reasoning error.

Structured Guideline from the `geometric_shapes` Task

Step 1: Parse and Map the SVG Path Commands

Execution: Systematically parse all SVG path commands (e.g., `M`, `L`, `A`). Record vertices, edges, and arcs. Treat `M` commands as boundaries for disconnected sub-paths.

Mistake: Misinterpreting `M` commands as continuous drawing instructions or including them as extra vertices.
Prevention: Explicitly isolate sub-paths introduced by `M`. Visually trace commands to confirm disjointness and segment boundaries.

Step 2: Identify Path Closure and Count Vertices/Edges

Execution: Check whether the final point matches the starting point. If closed, count distinct vertices and edges.

Mistake: Overlooking closure or miscounting repeated vertices, resulting in incorrect edge totals.

Prevention: Directly compare coordinates to verify closure. Use a deduplicated list of points to ensure accurate counting.

Step 3: Analyze Curved or Straight Line Features

Execution: Distinguish straight lines (`L`) from arcs (`A`). Extract arc parameters such as radii, sweep flags, and angles.

Mistake: Assuming all arcs are full circles or failing to recognize mixed shapes with curves and lines.

Prevention: Analyze arc direction, symmetry parameters (`rx`, `ry`), and flags to differentiate ellipses, sectors, and polygons.

Step 4: Validate Shape Properties Against Geometric Definitions

Execution: Match the figure to geometric definitions using vertex count, symmetry, angles, and side lengths (e.g., triangle, quadrilateral, ellipse).

Mistake: Prematurely classifying a figure without checking supporting properties (e.g., mislabeling a quadrilateral as a kite).

Prevention: Apply precise rules—verify symmetry, angle consistency, and equal-length sides where required.

Step 5: Apply a Process of Elimination to Confirm the Shape

Execution: Compare observed features against candidate shapes and eliminate those that do not match.

Mistake: Failing to rule out incompatible categories (e.g., misidentifying triangles as sectors due to curvature confusion).

Prevention: Use structured elimination (e.g., decision trees) to ensure all remaining options match key shape attributes.

Figure 9: A five-step structured guideline learned from the `geometric_shapes` task. Each step includes execution instructions, common mistake patterns, and prevention strategies to improve SVG-based shape reasoning.

Structured Reasoning on SVG Path with Step-wise Refinement

Input: `<path d="M 20,40 L 50,10 L 80,40 L 20,40" />`
Task: Identify the shape represented by the SVG path.
Options: (A) circle (B) heptagon (C) hexagon (D) kite (E) line (F) octagon (G) pentagon (H) rectangle (I) sector (J) triangle

Step 1: Parse and Map the SVG Path Commands
Result ✓: The path contains commands `M`, `L`, `L`, `L`, indicating a sequence of straight-line segments. The point sequence (20,40) → (50,10) → (80,40) → (20,40) suggests a continuous and closed path consisting of three connected segments.
Refined Result ✓: Same as above. No correction needed.

Step 2: Identify Path Closure and Count Vertices/Edges
Result ✗: The model failed to detect that the final point returns to the starting point, and incorrectly assumed the path was open. As a result, it counted only two line segments and classified the shape as a line (**E**).
Refined Result ✓: After recognizing the coordinate repetition, the path was confirmed to be closed. The vertex count was corrected to three, and the shape reclassified as **J: triangle**. *(based on **Mistake** and **Prevention**)*

Step 3: Analyze Curved or Straight Line Features
Result ✓: All commands involved were `L`, which represent straight-line segments in SVG path notation. No `A` arc commands were found. Therefore, the model deduced that the shape lacks curvature and consists solely of linear edges.
Refined Result ✓: Same as above. No correction needed.

Step 4: Validate Shape Properties Against Geometric Definitions
Result ✓: With three edges, no arcs, and no additional symmetry, the structure aligns with a triangle. The model ruled out quadrilaterals or curved forms by verifying vertex count and shape simplicity.
Refined Result ✓: Same as above. No correction needed.

Step 5: Apply a Process of Elimination to Confirm the Shape
Result ✓: The model excluded all shapes requiring more than three vertices (e.g., polygon, rectangle) and those requiring curvature (e.g., ellipse, sector). Triangle remained the only viable candidate.
Refined Result ✓: Same as above. No correction needed.

Final Option: After refinement, the model selects **(J) triangle** as the final answer.✓

Figure 10: Structured reasoning on an SVG path with intermediate refinement. An error in Step 2—failing to detect path closure—was successfully corrected via a refinement mechanism driven by the identified mistake and its prevention. All other steps were executed correctly.