Beyond Examples: High-level Automated Reasoning Paradigm in In-Context Learning via MCTS

Anonymous ACL submission

Abstract

In-context Learning (ICL) enables large language models to perform downstream tasks through advanced prompting and high-quality demonstrations. However, traditional ICL paradigms face limitations in complex reasoning tasks, primarily due to their reliance on example quality and the absence of explicit guidance signals. To address these challenges, this paper introduces HiAR-ICL, a **Hi**gh-level **A**utomated **R**easoning paradigm in **ICL** that shifts focus from specific examples to abstract reasoning patterns, thereby extending the conventional concept of 'context' in ICL. We define five atomic reasoning actions for constructing chain-structured reasoning patterns, then use Monte Carlo Tree Search to systematically explore reasoning paths and generate thought cards to guide subsequent inference. After that, we develop a cognitive complexity framework that dynamically matches test problems with appropriate thought cards. Experimental results demonstrate HiAR-ICL's effectiveness, achieving superior accuracy (80.6%) on the MATH benchmark with Qwen2.5-7B-Instruct, surpassing GPT-4o (76.6%).

1 Introduction

"Give a man a fish and you feed him for a day. Teach a man to fish and you feed him for a lifetime."
— An old proverb

Large language models (LLMs) have demonstrated remarkable capabilities across diverse tasks and domains (Zhao et al., 2023; OpenAI, 2023; Dubey et al., 2024). Their proficiency in complex reasoning, particularly in mathematical tasks, has emerged as a critical benchmark for evaluating fundamental cognitive abilities (Hao et al., 2023; Xi et al., 2024). Mastering multi-step reasoning requires rigorous adherence to intricate rules and sophisticated problem-solving strategies, presenting significant challenges for existing LLMs (Fu et al., 2023; Ahn et al., 2024; Plaat et al., 2024).

In-context learning (ICL) has emerged as a promising approach for enhancing LLMs' reasoning capabilities, distinguished by its simplicity and parameter-free nature (Zhou et al., 2024c). Rooted in analogy-based learning (Brown et al., 2020), ICL strategically employs curated demonstration examples to help models uncover latent patterns and generate sophisticated reasoning trajectories. Recent research has focused on improving ICL performance through advanced prompt engineering, including instruction optimization (Wang et al., 2023c) and demonstration selection (Luo et al., 2024). A key advancement is Chain-of-Thought (CoT) (Wei et al., 2022), which uses prompts like "Let's think step by step" to facilitate more structured problem-solving (Sprague et al., 2024).

Despite recent advances, current ICL paradigms face several limitations. First, empirical studies (Wang et al., 2023a; Cui et al., 2024; Wang et al., 2024d) reveal that LLMs' reasoning performance is highly sensitive to demonstration characteristics, including quantity, ordering, and label distributions. Suboptimal examples not only fail to maximize model performance but can impede reasoning abilities, often requiring considerable human curation. Second, existing ICL methods lack explicit guidance signals, forcing LLMs to infer implicit cues through imitation (Zhao et al., 2024). Third, ICL's generalization remains constrained. Reconstructing examples becomes necessary when encountering tasks with similar logical structures but varying presentation formats (Dong et al., 2024).

To address these challenges, we propose HiAR-ICL, a **Hi**gh-level **A**utomated **R**easoning paradigm in **ICL** through Monte Carlo Tree Search (MCTS). Our approach extends traditional ICL by redefining 'context' from mere examples to higher-order cognitive reasoning patterns. This paradigm embodies the principle of "teaching how to think, rather than what to think". Figure 1 compares the core concepts of our method with traditional ICL.

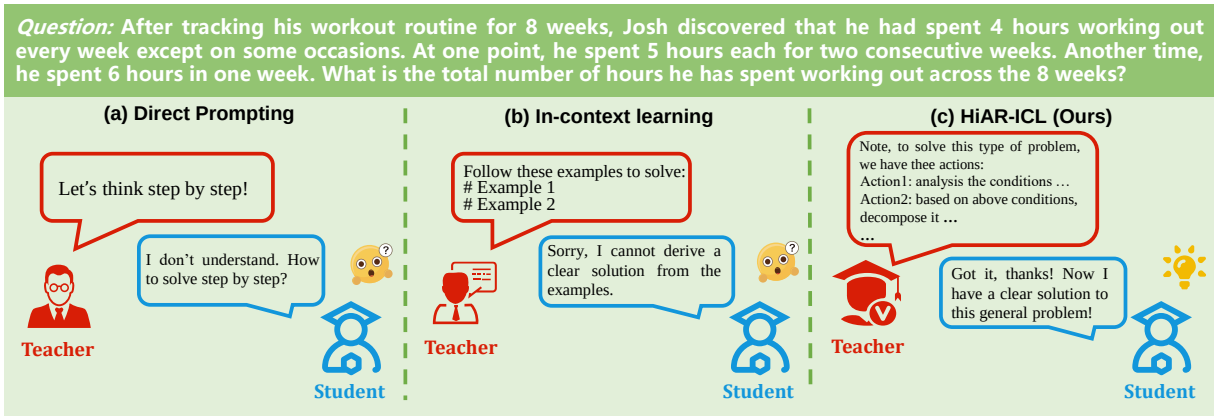


Figure 1: We compare HiAR-ICL with traditional zero-shot and few-shot in-context learning methods using a teacher-student paradigm. (a) Direct Prompting (Zero-shot CoT) provides a generic "Let's think step by step" instruction, which is insufficient for detailed step-by-step reasoning; (b) In-Context Learning (Few-shot CoT) offers carefully selected examples but struggles with dissimilar problems; (c) Our Method (HiAR-ICL) teaches high-level reasoning patterns, ensuring robust performance across various problems.

Specifically, our method comprises four key steps: (1) define atomic reasoning actions, (2) construct thought cards via MCTS, (3) select reasoning patterns, and (4) solve and verify. First, we define five atomic reasoning actions to form the building blocks of chain-structured reasoning patterns (termed "thought card"). These actions simulate human-like cognitive behaviors, including problem decomposition and reasoning step reflection. Second, leveraging a small set of randomly sampled seed data, we apply MCTS to derive reference reasoning patterns, constructing multiple thought cards. Third, we introduce a novel cognitive complexity framework with three metrics: subquestion count, problem condition complexity, and semantic similarity. This framework enables selecting the five optimal thought cards most closely aligned with the target problem's cognitive complexity. Finally, guided by the selected thought cards, we perform reasoning and validate the final solution through three verification mechanisms: outcome reward model, process reward model, and self-consistency checks. Extensive experiments demonstrate that HiAR-ICL significantly outperforms traditional ICL methods, even standing comparable to powerful closed-source models. Our main contributions are:

- **Novel ICL Insight**: Extending "context" in ICL from specific examples to higher-level cognitive reasoning patterns.
- **Automated Reasoning Paradigm**: Proposing an MCTS-based automated approach for generating and selecting optimal reasoning patterns.
- **Efficient Action-Chain Guided Reasoning**: Providing explicit guidance for each step of the reasoning process.
- **Superior Performance**: Achieving 80.6% accuracy on MATH with Qwen2.5-7B-Instruct, surpassing GPT-4o (76.6%).

2 Related Work

In-Context Learning via Examples In-context learning enables LLMs to learn from few demonstrations without fine-tuning (Zhou et al., 2024c; Dong et al., 2024). Approaches like CoT (Kojima et al., 2022; Wei et al., 2022) guide step-by-step reasoning, while Self-Consistency (Wang et al., 2023b) generates multiple reasoning paths to improve performance. However, these methods rely on example-level analogical learning with limited generalization and often require expert intervention for complex tasks (Wang et al., 2023a; Yang et al., 2024c; Zhao et al., 2024). In contrast, our approach shifts the focus to high-level reasoning patterns, enabling automated efficient inference across complex tasks without human intervention, even for smaller models under 10B parameters.

Tree-based Search LLMs exhibit impressive capabilities but face challenges in complex reasoning (Zhao et al., 2023). Tree search algorithms like MCTS (Chaslot et al., 2008) have emerged to expand search spaces (Świechowski et al., 2023) and enhance reasoning capabilities (Koh et al., 2024; Zhang et al., 2024b; Zhou et al., 2024a). Recent approaches such as Tree of Thought (Yao et al., 2023)

and GOT (Besta et al., 2024) explore non-linear reasoning paths through multiple LLM queries. While methods like AlphaMath (Chen et al., 2024) and rStar (Qi et al., 2024) advance reasoning capabilities, they often incur significant computational overhead. Our approach introduces a novel paradigm by frontloading computational resources and incorporating prior reasoning patterns, achieving competitive performance with lower complexity.

3 Methodology

Overview of HiAR-ICL This section introduces HiAR-ICL in detail. As shown in Figure 2, our approach consists of four main components:

- *Define Atomic Reasoning Actions*: Establish five human-like reasoning actions as building blocks for chain-structured thought cards.
- *Construct Thought Cards via MCTS*: Utilize MCTS to comprehensively construct cards.
- *Select Reasoning Patterns*: Identify optimal reasoning patterns based on the problem's cognitive complexity.
- *Solve and Verify*: Execute reasoning using selected thought cards and validate solutions through three verification mechanisms.

3.1 Define Atomic Reasoning Actions

Understanding human complex reasoning is crucial for modeling cognitive processes (Jaffe et al., 2023). Existing studies distinguish between two cognitive systems: System 1 and System 2 (Kahneman, 2011; Da Silva, 2023). While "System 1" represents fast, intuitive, yet error-prone thinking, "System 2" involves slow, deliberative thinking with superior performance. With the emergence of advanced models like OpenAI's o1, developing efficient "System 2" approaches to emulate human cognitive processes has gained significant research attention (Qin et al., 2024; Sun et al., 2024). Inspired by this, we introduce five atomic human-like reasoning actions to bridge the gap between model reasoning and human cognition: *System Analysis (SA, a_1)*, *One-Step Thought (OST, a_2)*, *Chain-of-Thought (CoT, a_3)*, *Divide and Conquer (DC, a_4)*, *Self-Reflection and Refinement (SRR, a_5)*. Detailed descriptions are provided in Appendix B.1.

3.2 Construct Thought Cards via MCTS

Following action definition, we can establish prior reasoning patterns. We introduce "*thought cards*", which serve as reasoning templates in subsequent reasoning (Section 3.4). Using a small seed dataset, we derive reasoning paths (Step 1) and distill them into multiple thought cards (Step 2). These cards serve as prior knowledge for test stage, facilitating efficient reasoning by providing reference insights.

Step 1: Acquire reasoning paths for seed data

As shown in Figure 2, we employ MCTS to iteratively evaluate and optimize the solution search process, generating high-quality reasoning paths for the seed dataset. This design combines MCTS's iterative nature with LLMs' inherent reasoning capabilities, leading to enhanced search outcomes (Ye et al., 2021; Zhou et al., 2024b). Following (Wang et al., 2024a; Qi et al., 2024), we formulate each complex reasoning question q as a tree search problem, where q represents the root node and subsequent nodes denote reasoning steps (actions and corresponding outcomes) generated by a policy LLM π_θ. We define the state S_{t-1} as the trajectory $q, s_1, ..., s_{t-1}$, where $S_0 = q$. The next step is sampled as $s_t \sim \pi_\theta(S_{t-1})$. Another critical component of MCTS is the reward function, which guides tree expansion. We define $Q(s)$ as the reward value for node s. Initially, all unexplored nodes are assigned $Q(s_i) = 0$. They will be updated using a weighted average of its current value and the Q-value of its child node:

$$Q_{new}(p) = (1-\alpha)Q_{old}(p) + \alpha Q(s) \quad (1)$$

where α is a discount factor, reflecting the importance of future rewards. For terminal nodes, following (Qi et al., 2024), we adopt the likelihood (confidence) of self-consistency majority voting as the reward value. This design eliminates the need for external supervision, thereby enhancing the generalization of our approach.

Specifically, this step comprises 4 MCTS phases:

(1) Selection. This phase identifies optimal nodes for expansion. Starting from the root node, a child node is chosen at each tree level until reaching a leaf node, which we define as either achieving maximum tree depth or arriving at an answer here. To balance the exploration and exploitation, we employ the well-known Upper Confidence Bounds applied to Trees (UCT) (Kocsis and Szepesvári, 2006) for node selection:

$$UCT(s) = Q(s) + w\sqrt{\frac{lnN(p)}{N(s)}} \quad (2)$$

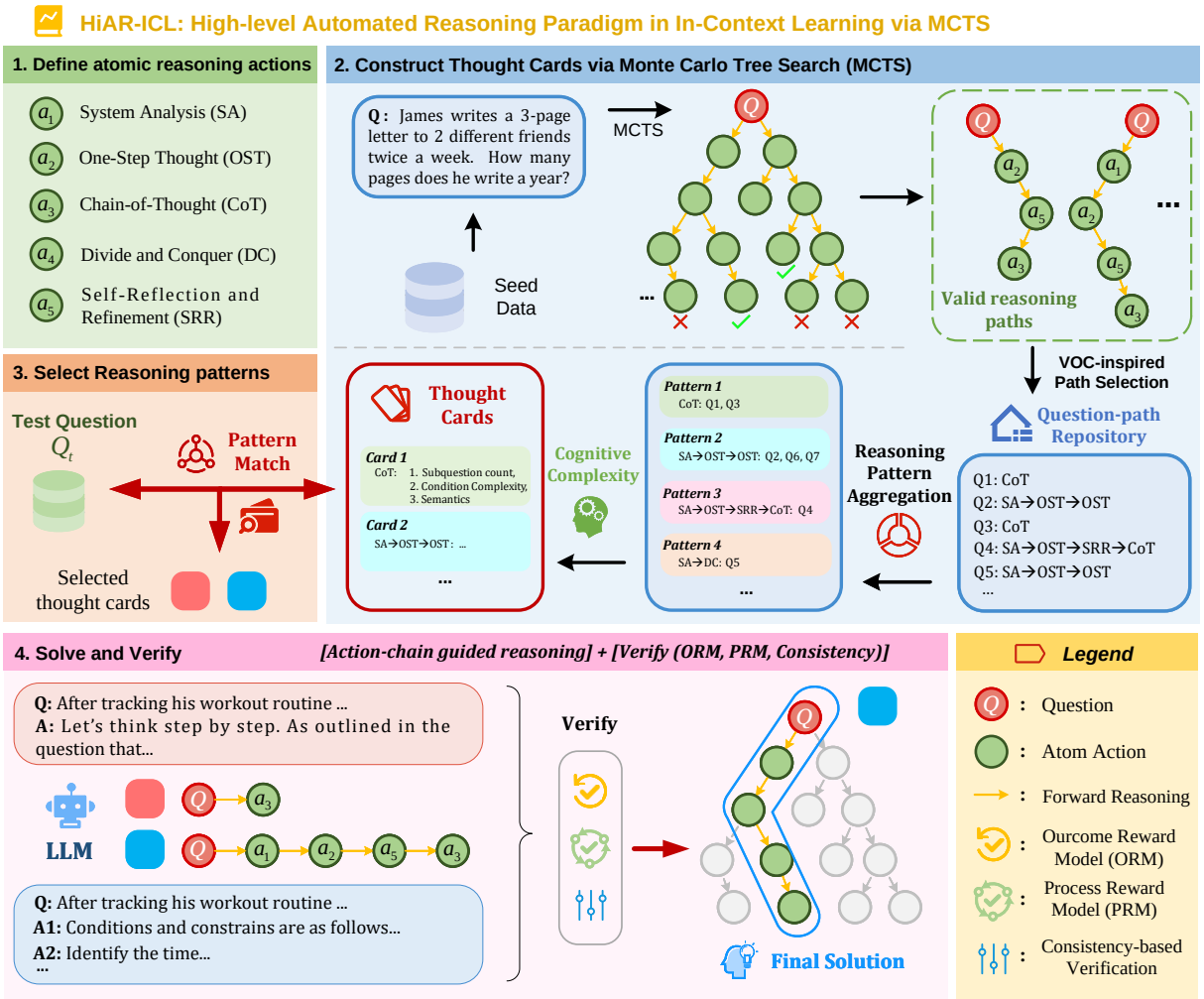


Figure 2: Flowchart of our method HiAR-ICL. This framework consists of four main parts: (1) Define Atomic Reasoning Actions; (2) Construct Thought Cards via MCTS; (3) Select Reasoning Patterns; (4) Solve and Verify.

where $Q(s)$ is the reward value for node s, $N(s)$ is the visit count, p is the parent node, and w is the exploration weight. The node with the highest UCT value is selected for subsequent phases.

(2) Expansion. During the expansion phase, the selected node s is expanded by sampling n actions from π_θ and then generating corresponding reasoning outcomes, as described above. This results in n child nodes, which are added to the tree and stored in an external long-term memory structure.

(3) Simulation. A complete simulation involves iteratively selecting and expanding nodes from the initially selected node until reaching a terminal node. A terminal node is defined as either the maximum tree depth or an answer node, marked by a specific answer indicator ("the answer is" in this paper). At each tree level, nodes are sampled and expanded following a consistent procedure.

To enhance efficiency and avoid unnecessary expansion, we implement an early termination strategy. Inspired by (Fluri et al., 2023; Zhou et al., 2024b), we incorporate a heuristic based on self-consistency (Wang et al., 2023b). This strategy exploits the observation that actions sampled multiple times at the same state are more likely to be accurate, indicating that the model has likely completed the task successfully. Specifically, if the model's consistency score exceeds a predefined threshold c, i.e., $SC(s) > c$, the simulation terminates early.

(4) Backpropagation. Upon reaching the rollout end, node information is updated along the simulation path $s_0, ...s_d$. Visit counts are incremented ($N_{new}(s) = N_{old}(s) + 1$), and node value $Q(s)$ is propagated backward to its parent node p using Eq. 1. These updated values are used in the UCT formula (Eq. 2) to guide node selection.

Step 2: Distill paths into thought cards

After executing MCTS, we obtain a tree structure for each seed dataset question, yielding multi-

ple valid reasoning paths. To identify the optimal reasoning trajectory, we draw inspiration from the concept of Value of Computation (VOC) (Russell and Wefald, 1991). VOC posits that intelligent systems should optimize the trade-off between computational benefits and costs. Accordingly, we propose a novel VOC-inspired selection metric:

$$score(x, pa, y) = k \cdot R(pa|x) - (1 - k) \cdot C(pa) \tag{3}$$

where x is the task input, y is the target answer, pa represents a candidate reasoning path, k balances computational benefits against costs. $R(pa|x)$ denotes the path's final reward score, while $C(pa)$ measures the reasoning sequence length. For simplicity, we define $Reward(pa|x)$ as the Q-value of the leaf node and $C(pa)$ as the count of utilized actions in the reasoning sequence. More complex settings are left for future research.

Then, for each question in the seed dataset, we select the path with highest $score(x, pa, y)$ to build a *Question-path repository* with one-to-one mappings. Inspired by metareasoning principles (Russell and Wefald, 1991; De Sabbata et al., 2024), which advocate for adaptive reasoning strategies, we distill these question-path pairs into abstract thought cards for test-time reference. This transformation is guided by a novel cognitive complexity framework designed for complex reasoning (Lee and Heyworth, 2000; Embretson and Daniel, 2008). This framework encompasses three dimensions: (1) *Subquestion Count (SC)*: Quantifies the number of subproblems; (2) *Problem Condition Complexity (PCC)*: Measures known condition count; (3) *Semantic Similarity (SS)*: Evaluates semantic distance between target and seed problems.

As shown in Figure 2, using SC and PCC metrics, we refine the repository into high-level *thought cards*, reserving SS for the testing phase. The next section describes how we utilize these thought cards to provide reasoning guidance signals.

3.3 Select Reasoning Patterns

During testing, we compute SC, PCC, and SS for a test question q_t using Llama3-8B-Instruct. The SS metric is based on the semantic distance to the seed dataset. We then match these metrics to our pre-existing thought cards, selecting three to five cards that align most closely with the test question q_t. These chosen cards serve as reference templates for q_t in the final solution generation and verification.

3.4 Solve and Verify

Solve Using selected thought cards as templates, we generate candidate solutions for a given test problem q_t. By incorporating prior reasoning patterns, our method extends the tree-structured reasoning framework and eliminates the need for extensive node expansion. Following the specified action sequences on thought cards (e.g. $SA \rightarrow OST \rightarrow CoT$), as illustrated in Figure 2, we can derive multiple candidate solutions for q_t.

Verify Identifying the most accurate reasoning trajectory among multiple candidate solutions represents a critical challenge (Uesato et al., 2022; Qi et al., 2024; Zhang et al., 2024a). We investigate three verification methods: process-supervision, outcome-supervision, and self-consistency. Empirical results show that even simple self-consistency effectively identifies precise reasoning chains.

In summary, our approach can also be seen as an optimized variant of tree search algorithms. Unlike traditional MCTS methods that require extensive and computationally expensive exploration, our method strategically reduces computational complexity. By incorporating prior knowledge, HiAR-ICL significantly enhances search efficiency. As shown in Figure 2, our approach enables efficient tree traversal along promising trajectories, maintaining high performance while reducing computational complexity. This may offer valuable insights into developing efficient reasoning strategies.

4 Experimental Settings

Datasets To evaluate HiAR-ICL, we conduct experiments across diverse datasets and reasoning tasks. We randomly sample 200 instances from the training sets as seed data for thought card construction (Section 3.2) and assess performance on corresponding test sets. Our evaluation benchmarks include: (1) arithmetic reasoning: GSM8K_{1319} (Cobbe et al., 2021) and SVAMP_{300} (Patel et al., 2021); (2) mathematical reasoning: MATH_{500} (Hendrycks et al., 2021); (3) commonsense reasoning: StrategyQA_{687} (Geva et al., 2021).

Baselines We evaluate HiAR-ICL against three strong baseline categories: (1) traditional example-based ICL methods, including zero-shot CoT (Kojima et al., 2022), few-shot CoT (Wei et al., 2022), and CoT+SC (Wang et al., 2023b); (2) tree-based methods, including ToT (Yao et al., 2023), RAP (Hao et al., 2023), ReST-MCTS* (Zhang et al.,

Table 1: Evaluation of HiAR-ICL's reasoning performance across four benchmarks, with the best results in each box highlighted in **bold**. HiAR-ICL consistently outperforms other methods.

Model	Method	Mathematics	Arithmetic		Commense	Average
		MATH	GSM8K	SVAMP	StrategyQA	
Qwen2.5-14B-instruct (Qwen Team, 2024)	Zero-shot CoT	69.8	92.4	91.6	62.8	79.1
	Few-shot CoT	80.0	94.8	91.3	53.1	79.8
	CoT+SC@4	76.2	94.0	91.0	69.7	82.7
	Ours	**81.4**	**95.8**	**93.7**	**77.3**	**87.0**
Qwen2.5-7B-instruct (Qwen Team, 2024)	Zero-shot CoT	64.8	86.2	91.3	52.8	73.7
	Few-shot CoT	68.6	91.6	92.3	67.6	80.1
	CoT+SC@4	76.4	92.0	92.3	73.2	83.4
	Ours	**80.6**	**93.7**	**93.0**	**76.0**	**85.9**
Qwen2-7B-instruct (Yang et al., 2024a)	Zero-shot CoT	36.9	76.6	85.2	55.3	63.5
	Few-shot CoT	52.9	85.7	87.3	62.3	72.0
	CoT+SC@4	55.6	87.7	90.3	65.5	74.8
	Ours	**66.8**	**91.8**	**92.7**	**72.0**	**80.9**
Yi-1.5-6B-Chat (Young et al., 2024)	Zero-shot CoT	30.4	76.4	64.4	46.2	54.3
	Few-shot CoT	40.5	78.9	81.3	61.1	65.4
	CoT+SC@4	42.2	79.4	87.6	65.2	68.6
	Ours	**57.4**	**86.4**	**91.3**	**70.3**	**76.4**
Llama3-8B-Instruct (Dubey et al., 2024)	Zero-shot CoT	5.8	68.3	70.9	57.2	50.5
	Few-shot CoT	17.8	74.5	81.0	68.4	60.4
	CoT+SC@4	28.8	80.6	88.0	66.8	66.0
	Ours	**46.6**	**89.6**	**92.7**	**73.4**	**75.6**
Llama3.1-8B-Instruct (Meta AI, 2024)	Zero-shot CoT	18.0	61.5	69.3	52.4	50.3
	Few-shot CoT	47.2	76.6	82.0	63.6	67.3
	CoT+SC@4	44.2	80.5	85.6	69.8	70.0
	Ours	**58.0**	**90.7**	**93.0**	**75.7**	**79.4**

2024a), LiteSearch (Wang et al., 2024a), BEATS (Sun et al., 2024), LLaMA-Berry (Zhang et al., 2024c), and rStar (Qi et al., 2024). (3) powerful closed-source LLMs, including GPT-4 (OpenAI, 2023), GPT-4o (OpenAI, 2024), Claude-3.5 (Anthropic, 2024a) and Gemini-1.5-pro (Google DeepMind, 2024).

Evaluation Metrics We evaluate our approach using two metrics: *accuracy*, based on the strict matching between the model's final answer and ground truth, and *average inference time* to assess computational efficiency compared to existing search-based approaches.

5 Results and Discussion

This section validates the effectiveness of HiAR-ICL from four aspects: 5.1 performance, 5.2 computational efficiency, 5.3 out-of-distribution generalization, and 5.4 ablation studies.

5.1 Performance Comparison with Baselines

Results on Diverse Reasoning Benchmarks Table 1 presents the performance of HiAR-ICL across four mainstream reasoning benchmarks. We have two important findings:

◇ HiAR-ICL consistently outperforms traditional ICL methods across all tasks. For example, Llama3-8B-Instruct's accuracy on the MATH benchmark improved from 17.8% (few-shot CoT) to 46.6% with HiAR-ICL, a 2.6x improvement.

◇ The method yields substantial improvements on small language models, with Qwen2-7B-Instruct increasing from 52.9% to 66.8% and Yi-1.5-6B-chat from 40.5% to 57.4%. These results highlight HiAR-ICL's potential for small models.

Comparison with Closed-Source LLMs To demonstrate HiAR-ICL's effectiveness, we compare HiAR-ICL with leading closed-source models and large open-source models. As shown in Figure 3, HiAR-ICL-empowered LLMs achieve competitive performance, matching models with hundreds of billions of parameters and even surpassing powerful closed-source models. Notably, the 7B Qwen2.5 model achieves 80.6% accuracy on the challenging MATH benchmark, outperforming GPT-4o (76.6%). Similarly, employing our rea-

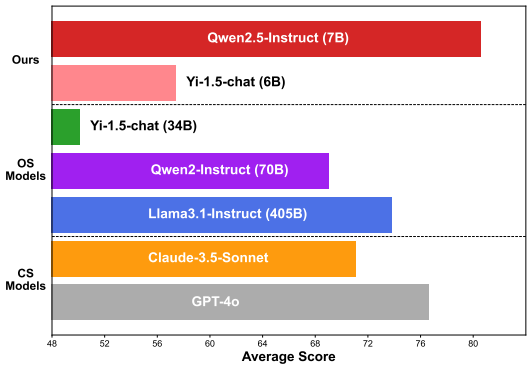
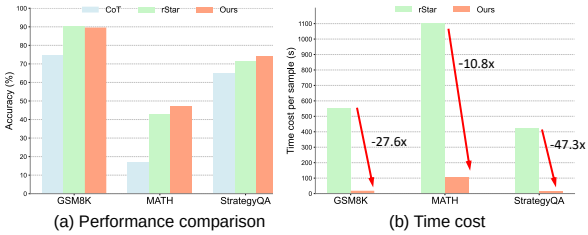


Figure 3: Comparison between HiAR-ICL and other powerful LLMs on the challenging MATH dataset. 'OS' and 'CS' denote open-source and closed-source models.



(a) Performance comparison (b) Time cost

Figure 4: Comparison with SOTA tree-based method, rStar. (a) Performance Comparison. (b) Time Cost.

soning paradigm, Yi-1.5-6B surpasses Yi-1.5-34B. Detailed results are shown in Appendix D.1.

Comparison with Tree-based Methods We compare our algorithm with other tree-based reasoning methods in Table 2. While existing methods like ToT and RAP face significant challenges, our method demonstrates superior performance and notable generalization across various models and datasets. The performance gap widens with increasing reasoning complexity from GSM8K to MATH. Specifically, on Llama3-8B-Instruct, our method achieves 46.6% accuracy compared to the best baseline's 42.9%, while on Yi-1.5-6B-Chat, it improves from 61.5% to 66.8%. While traditional tree search methods provide broad exploration capabilities, they struggle to identify precise reasoning paths, particularly for complex reasoning tasks. Our approach addresses this limitation by leveraging prior thought cards to effectively guide reasoning based on problem complexity.

5.2 Computational Efficiency

We further compare with rStar (Qi et al., 2024), a recently proposed method that significantly expands the solution search space and achieves state-of-the-art performance in tree-based reasoning. Our analysis in Figure 4 systematically assesses performance and computational efficiency. We observe that HiAR-ICL achieves competitive performance with rStar while substantially reducing time complexity. Specifically, we achieve time reductions of 27.6x on GSM8K and 47.3x on StrategyQA through adaptive pattern selection. On the more challenging MATH dataset, our method maintains a tenfold speedup, demonstrating efficient and adaptive reasoning across problem complexities.

Table 2: Performance of different tree-based methods. Our results are marked in blue, with baseline results sourced from the original paper when accessible. The best results are highlighted in **bold**.

Model	Method	MATH	GSM8K
Yi-1.5-6B-Chat	BEATS	51.2	76.1
	Ours	**57.4**	**86.4**
Qwen2-7B-instruct	BEATS	61.5	83.0
	Ours	**66.8**	**91.8**
Llama3-8B-Instruct	ToT	13.6	69.0
	RAP	18.8	80.5
	ReST-MCTS*	34.2	-
	LiteSearch	-	82.3
	LLaMA-Berry	39.6	88.1
	BEATS	42.9	88.4
	Ours	**46.6**	**89.6**
Llama3.1-8B-Instruct	LLaMA-Berry	54.8	89.8
	Ours	**58.0**	**90.7**

5.3 Out-of-Distribution Generalization

Recent studies have highlighted the critical impact of distributional bias on the reliability of LLMs (Yuan et al., 2023; Wang et al., 2024b). Despite impressive in-distribution (ID) performance, these models substantially underperform when confronted with out-of-distribution (OOD) data (Berglund et al., 2024; Yang et al., 2024b). This challenge is compounded by the inherent difficulty of acquiring sufficient training data.

In this paper, we evaluate HiAR-ICL's performance against in-context learning (ICL) and supervised fine-tuning (SFT) under OOD scenarios. To ensure a fair comparison, we use the same seed data from Section 3.2 for SFT. As illustrated in Figure 5, while ICL and SFT experience significant performance degradation, HiAR-ICL demonstrates remarkable resilience, preserving robust performance across multiple models and datasets. These results underscore HiAR-ICL's superior robustness and generalization capabilities, positioning it as a more reliable and adaptable solution for handling diverse reasoning tasks, including both ID and OOD data.

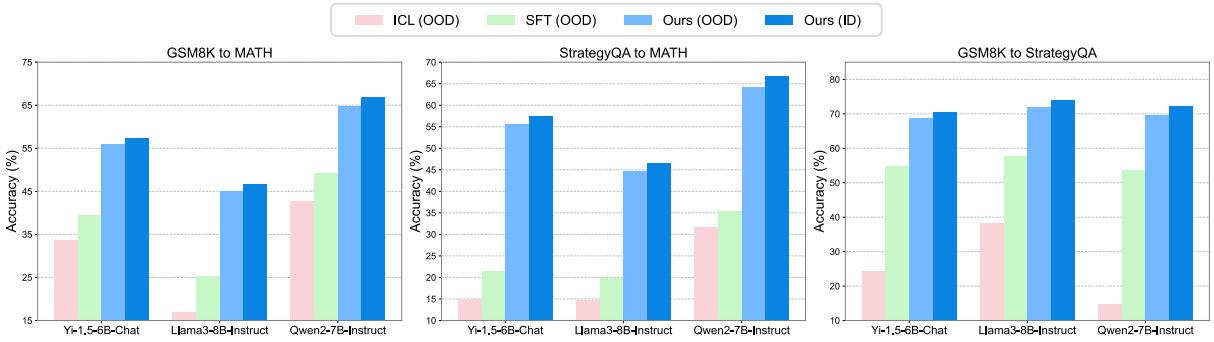


Figure 5: Cross-domain performance comparison. 'ID' and 'OOD' represent in-distribution and out-of-distribution.

Table 3: Ablation results on Llama3-8B-Instruct. HiAR-ICL requires every component for optimal performance.

Model Setting	Module	MATH	GSM8K	SVAMP	StrategyQA	Average	△ (↓)
HiAR-ICL	-	**46.6**	**89.6**	**92.7**	**73.4**	**75.6**	-
– w/o thought cards (random actions)	*Sec. 3.2*	33.4	79.9	84.3	63.3	65.2	-10.4
– w/o card match (random card)	*Sec. 3.3*	39.2	86.1	89.0	68.0	70.6	-5.0
– w/ card match (semantic max)	*Sec. 3.3*	42.0	86.9	90.0	71.5	72.6	-3.0
– w/ card match (semantic average)	*Sec. 3.3*	40.2	86.4	89.0	70.6	71.5	-4.1
– w/o verification (random selection)	*Sec. 3.4*	40.2	83.3	90.0	69.6	70.8	-4.8
– w/ verification (SC)	*Sec. 3.4*	42.6	87.3	92.0	70.3	73.1	-2.5
– w/ verification (ORM)	*Sec. 3.4*	43.8	88.8	91.7	73.4	74.4	-1.2

5.4 Ablation Studies and Analysis

Effect of Thought Cards HiAR-ICL constructs thought cards to provide high-level reasoning patterns for test-time inference. As shown in Table 3 (Row 2), replacing the thought cards construction module with randomly sampled action-based reasoning chains (path lengths 2-5) leads to a 10.4% accuracy drop. This finding highlights the importance of precise priori knowledge in improving reasoning performance and the risk of model capability degradation with imprecise information.

Effect of Cognitive Complexity Framework Given off-the-shelf thought cards, if the cards are not matched based on question complexity but are instead randomly selected as guiding information, we observe a performance drop of 5.0% in Table 3 (Rows 3). This indicates that matching thought cards to the complexity of the question is crucial for optimal performance.

Effect of Verification Module To evaluate the effectiveness of the verification module, we consider four configurations: random solution selection, Self-Consistency (SC), Outcome Reward Model (ORM), and Process Reward Model (PRM). As shown in Table 3 (Rows 6-8), path selection plays a critical role, with PRM (default in this paper, HiAR-ICL) achieving the highest performance. The simple yet effective SC method also performs well. Even random selection shows reasonable performance, with only a slight accuracy drop of 4.8%. This also implies that our preliminary inference generates generally accurate candidate solutions, confirming the effectiveness of the earlier modules.

Other Analysis Finally, we include some additional results and analysis in the Appendix, including a more comprehensive performance comparison (D.1), integration with SFT (D.2), performance across varying problem complexity (D.3), test-time scaling law (D.4), and case studies (F).

6 Conclusion

We propose HiAR-ICL, a novel automated reasoning paradigm that extends in-context learning by incorporating abstract reasoning patterns, enabling LLMs to develop genuine reasoning capabilities beyond mere example imitation. Our approach transcends traditional example-based methods and search techniques, demonstrating significant performance improvements in complex reasoning tasks. By shifting focus from human-curated examples to adaptive reasoning strategies, HiAR-ICL establishes a promising foundation for advancing complex reasoning in LLMs, opening new avenues for automated reasoning research.

Limitations

Despite HiAR-ICL's promising performance, it has three limitations. First, current experimental validation is constrained to a narrow range of reasoning tasks, primarily focusing on arithmetic, mathematical, and commonsense reasoning. This limited evaluation scope restricts our understanding of HiAR-ICL's broader applicability. Future research will address this by expanding the experimental framework to include a more diverse set of reasoning challenges, such as symbolic logic and multimodal reasoning tasks.

Secondly, the five fundamental actions defined, currently grounded in the limitations of existing reasoning literature (detailed in Appendix B.1), may face generalization challenges. Future research will focus on developing more adaptive and intelligent approaches for atomic action definition, with the aim of enhancing the flexibility and robustness of the HiAR-ICL paradigm.

Lastly, during the testing phase, reasoning path selection is determined by problem complexity metrics defined in the paper, which may impose limitations. For example, in multimodal reasoning domains involving both visual and textual information, selecting the appropriate reasoning path remains a challenge. We aim to develop more generalized and flexible path selection strategies to address these limitations.

References

Bo Adler, Niket Agarwal, Ashwath Aithal, Dong H Anh, Pallab Bhattacharya, Annika Brundyn, Jared Casper, Bryan Catanzaro, Sharon Clay, Jonathan Cohen, et al. 2024. Nemotron-4 340b technical report. *arXiv preprint arXiv:2406.11704*.

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 225–237, St. Julian's, Malta. Association for Computational Linguistics.

Anthropic. 2024a. Introducing claude 3.5 sonnet.

Anthropic. 2024b. Introducing the next generation of claude.

Edward Beeching, Shengyi Costa Huang, Albert Jiang, Jia Li, Benjamin Lipkin, Zihan Qina, Kashif Rasul, Ziju Shen, Roman Soletskyi, and Lewis Tunstall. 2024. Numinamath 72b cot. https://huggingface.co/AI-MO/NuminaMath-72B-CoT.

Lukas Berglund, Meg Tong, Maximilian Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. 2024. The reversal curse: LLMs trained on "a is b" fail to learn "b is a". In *The Twelfth International Conference on Learning Representations*.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. 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.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

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.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. 2024. Alphamath almost zero: Process supervision without process. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Sijia Chen and Baochun Li. 2024. Toward adaptive reasoning in large language models with thought rollback. In *Forty-first International Conference on Machine Learning*.

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

Yingqian Cui, Pengfei He, Xianfeng Tang, Qi He, Chen Luo, Jiliang Tang, and Yue Xing. 2024. A theoretical understanding of chain-of-thought: Coherent reasoning and error-aware demonstration. *arXiv preprint arXiv:2410.16540*.

Sergio Da Silva. 2023. System 1 vs. system 2 thinking. *Psych*, 5(4):1057–1076.

C Nicolò De Sabbata, Theodore R Sumers, and Thomas L Griffiths. 2024. Rational metareasoning for large language models. *arXiv preprint arXiv:2410.05563*.

Hao Dong, Zihan Ding, and Shanghang Zhang. 2020. *Deep Reinforcement Learning: Fundamentals, Research and Applications*, 1 edition, volume 1 of *eBook Packages: Mathematics and Statistics*. Springer Singapore.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. A survey on in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1107–1128, Miami, Florida, USA. Association for Computational Linguistics.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Susan E Embretson and Robert C Daniel. 2008. Understanding and quantifying cognitive complexity level in mathematical problem solving items. *Psychology Science*, 50(3):328.

Lukas Fluri, Daniel Paleka, and Florian Tramèr. 2023. Evaluating superhuman models with consistency checks. In *Socially Responsible Language Modelling Research*.

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing smaller language models towards multi-step reasoning. In *International Conference on Machine Learning*, pages 10421–10430. PMLR.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Google DeepMind. 2024. Gemini models.

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, Singapore. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

P. I. Jaffe, R. A. Poldrack, R. J. Schafer, and et al. 2023. Modelling human behaviour in cognitive tasks with latent dynamical systems. *Nature Human Behaviour*, 7:986–1000.

Daniel Kahneman. 2011. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, New York, NY.

Jikun Kang, Xin Zhe Li, Xi Chen, Amirreza Kazemi, Qianyi Sun, Boxing Chen, Dong Li, Xu He, Quan He, Feng Wen, et al. 2024. Mindstar: Enhancing math reasoning in pre-trained llms at inference time. *arXiv preprint arXiv:2405.16265*.

Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, pages 282–293, Berlin, Heidelberg. Springer Berlin Heidelberg.

Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. 2024. Tree search for language model agents. *arXiv preprint arXiv:2407.01476*.

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.

Fong-Lok Lee and Rex Heyworth. 2000. Problem complexity: A measure of problem difficulty in algebra by using computer. *EDUCATION JOURNAL-HONG KONG-CHINESE UNIVERSITY OF HONG KONG-*, 28(1):85–108.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5315–5333, Toronto, Canada. Association for Computational Linguistics.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*.

Sheng Lu, Irina Bigoulaeva, Rachneet Sachdeva, Harish Tayyar Madabushi, and Iryna Gurevych. 2023. Are emergent abilities in large language models just in-context learning? *arXiv preprint arXiv:2309.01809*.

Man Luo, Xin Xu, Yue Liu, Panupong Pasupat, and Mehran Kazemi. 2024. In-context learning with retrieved demonstrations for language models: A survey. *Transactions on Machine Learning Research*. Survey Certification.

Meta AI. 2024. Llama: Latest models.

Mistral AI team. 2024a. Cheaper, better, faster, stronger.

Mistral AI team. 2024b. Large enough.

OpenAI. 2022. Introducing chatgpt.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

OpenAI. 2024. Hello gpt-4o.

Arkil Patel, S. Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? In *North American Chapter of the Association for Computational Linguistics*.

Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki van Stein, and Thomas Back. 2024. Reasoning with large language models, a survey. *arXiv preprint arXiv:2407.11511*.

Zhenting Qi, Mingyuan Ma, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. 2024. Mutual reasoning makes smaller llms stronger problem-solvers. *arXiv preprint arXiv:2408.06195*.

Yiwei Qin, Xuefeng Li, Haoyang Zou, Yixiu Liu, Shijie Xia, Zhen Huang, Yixin Ye, Weizhe Yuan, Hector Liu, Yuanzhi Li, et al. 2024. O1 replication journey: A strategic progress report–part 1. *arXiv preprint arXiv:2410.18982*.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Stuart Russell and Eric Wefald. 1991. Principles of metareasoning. *Artificial Intelligence*, 49(1):361–395.

Zayne Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. 2024. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning. *arXiv preprint arXiv:2409.12183*.

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*.

Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. 2023. Monte carlo tree search: A review of recent modifications and applications. *Artificial Intelligence Review*, 56(3):2497–2562.

Ye Tian, Baolin Peng, Linfeng Song, Lifeng Jin, Dian Yu, Haitao Mi, and Dong Yu. 2024. Toward self-improvement of llms via imagination, searching, and criticizing. *arXiv preprint arXiv:2404.12253*.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

Ante Wang, Linfeng Song, Ye Tian, Baolin Peng, Dian Yu, Haitao Mi, Jinsong Su, and Dong Yu. 2024a. Litesearch: Efficacious tree search for llm. *arXiv preprint arXiv:2407.00320*.

Jindong Wang, Xixu Hu, Wenxin Hou, Hao Chen, Runkai Zheng, Yidong Wang, Linyi Yang, Wei Ye, Haojun Huang, Xiubo Geng, Binxing Jiao, Yue Zhang, and Xing Xie. 2024b. On the robustness of chatgpt: An adversarial and out-of-distribution perspective. *IEEE Data Eng. Bull.*, 47(1):48–62.

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023a. Label words are anchors: An information flow perspective for understanding in-context learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9840–9855, Singapore. Association for Computational Linguistics.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024c. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, Bangkok, Thailand. Association for Computational Linguistics.

Song Wang, Zihan Chen, Chengshuai Shi, Cong Shen, and Jundong Li. 2024d. Mixture of demonstrations for in-context learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023c. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

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

Siwei Wu, Zhongyuan Peng, Xinrun Du, Tuney Zheng, Minghao Liu, Jialong Wu, Jiachen Ma, Yizhi Li, Jian Yang, Wangchunshu Zhou, et al. 2024. A comparative study on reasoning patterns of openai's o1 model. *arXiv preprint arXiv:2410.13639*.

Zhiheng Xi, Wenxiang Chen, Boyang Hong, Senjie Jin, Rui Zheng, Wei He, Yiwen Ding, Shichun Liu, Xin

Guo, Junzhe Wang, Honglin Guo, Wei Shen, Xiaoran Fan, Yuhao Zhou, Shihan Dou, Xiao Wang, Xinbo Zhang, peng sun, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Training large language models for reasoning through reverse curriculum reinforcement learning. In *Forty-first International Conference on Machine Learning*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Haoran Yang, Yumeng Zhang, Jiaqi Xu, Hongyuan Lu, Pheng-Ann Heng, and Wai Lam. 2024b. Unveiling the generalization power of fine-tuned large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 884–899, Mexico City, Mexico. Association for Computational Linguistics.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E Gonzalez, and Bin Cui. 2024c. Buffer of thoughts: Thought-augmented reasoning with large language models. *Advances in Neural Information Processing Systems*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822. Curran Associates, Inc.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. 2021. Mastering atari games with limited data. In *Advances in Neural Information Processing Systems*, volume 34, pages 25476–25488. Curran Associates, Inc.

Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, et al. 2024. Yi: Open foundation models by 01. ai. *arXiv preprint arXiv:2403.04652*.

Lifan Yuan, Yangyi Chen, Ganqu Cui, Hongcheng Gao, FangYuan Zou, Xingyi Cheng, Heng Ji, Zhiyuan Liu, and Maosong Sun. 2023. Revisiting out-of-distribution robustness in NLP: Benchmarks, analysis, and LLMs evaluations. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024a. Rest-mcts*: Llm self-training via process reward guided tree search. *Advances in Neural Information Processing Systems*.

Di Zhang, Jiatong Li, Xiaoshui Huang, Dongzhan Zhou, Yuqiang Li, and Wanli Ouyang. 2024b. Accessing gpt-4 level mathematical olympiad solutions via monte carlo tree self-refine with llama-3 8b. *arXiv preprint arXiv:2406.07394*.

Di Zhang, Jianbo Wu, Jingdi Lei, Tong Che, Jiatong Li, Tong Xie, Xiaoshui Huang, Shufei Zhang, Marco Pavone, Yuqiang Li, et al. 2024c. Llama-berry: Pairwise optimization for o1-like olympiad-level mathematical reasoning. *arXiv preprint arXiv:2410.02884*.

Anhao Zhao, Fanghua Ye, Jinlan Fu, and Xiaoyu Shen. 2024. Unveiling in-context learning: A coordinate system to understand its working mechanism. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 12375–12400, Miami, Florida, USA. Association for Computational Linguistics.

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

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2024a. Language agent tree search unifies reasoning, acting, and planning in language models. In *Forty-first International Conference on Machine Learning*.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2024b. Language agent tree search unifies reasoning, acting, and planning in language models. In *Forty-first International Conference on Machine Learning*.

Yuxiang Zhou, Jiazheng Li, Yanzheng Xiang, Hanqi Yan, Lin Gui, and Yulan He. 2024c. The mystery of in-context learning: A comprehensive survey on interpretation and analysis. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14365–14378, Miami, Florida, USA. Association for Computational Linguistics.

Appendix of HiAR-ICL

This supplementary material provides in-depth insights into our HiAR-ICL method, covering additional descriptions, experimental details, and results. The appendix is organized as follows:

Table of Contents

A. Preliminaries

- A.1. Overall Notations
- A.2. LLM Reasoning
- A.3. In-Context Learning
- A.4. Monte Carlo Tree Search
- A.5. Verification Methods

B. Algorithm Details

- B.1. Action Spaces
- B.2. Reward Value in MCTS
- B.3. Thought Cards Selection
- B.4. Verification

C. Experimental Settings

- C.1. Models
- C.2. Datasets
- C.3. Baselines
- C.4. Evaluation Details
- C.5. Implementation Details

D. Additional Experiments

- D.1. Comparison with Powerful LLMs
- D.3. Performance across Complexity
- D.2. Integration with SFT
- D.4. Ablation Studies

E. Prompts

F. Case Studies

A Preliminaries

This section describes notations (A.1), LLM reasoning (A.2), in-context learning (A.3), monte carlo tree search (A.4), and verification methods (A.5).

A.1 Overall Notations

The definitions for notations are in Table 4.

A.2 LLM Reasoning

LLMs have shown impressive performance across various reasoning tasks (Zhao et al., 2023), including mathematical (Ahn et al., 2024) and commonsense reasoning (Plaat et al., 2024). Appropriate reasoning methods can substantially enhance LLM problem-solving capabilities, potentially transforming small models into powerful problem-solvers (Fu et al., 2023; Qi et al., 2024; Chen and Li, 2024). Given a policy model, π_θ (an autoregressive pre-trained LLM) and an input problem x, π can autogressive generate an output sequence $output = (s_0, s_1, s_2, ..., s_T)$ by predicting the next token, where $s_0 := x$ and $ans_p = s_T$. Each output sequence $(s_0, s_1, s_2, ..., s_T)$ is termed a reasoning trajectory $traj$. The conditional probability distribution of generating the complete reasoning trajectory is:

$$\pi(traj \mid x) = \prod_{t=1}^{T} \pi\left(s_t \mid x, s_{<t}\right) \tag{4}$$

Following (Hao et al., 2023; Qi et al., 2024), we can conceptualize LLMs as world models, with the complex reasoning process formulated as a Markov decision process. Specifically, when addressing complex reasoning challenges in real-world scenarios, at each time step t, the model receives a state S_{t-1}, comprising the original input problem x and preceding reasoning steps $(s_0, s_1, s_2, ..., s_{t-1})$. The policy model π_θ then generates the current action $a_t = \pi_\theta(\Phi(S_{t-1}))$, which prompts the LLM to produce the next reasoning step s_t. The entire process, from the initial step s_0 to the final output s_T, naturally forms a complete trajectory or chain of thought.

Recent research has focused on developing diverse methods to enhance LLMs' reasoning capabilities, including zero-shot prompting, few-shot prompting, chain-of-thought (CoT), tree-of-thought (ToT), and Monte Carlo tree search (MCTS). These approaches aim to improve overall performance through the following formulation:

$$P_\pi\left(y_p = y_g \mid Q\right) = \mathbb{E}_{(s_0, s_1, \cdots, s_T) \sim P_\pi(traj|x)} \left[P\left(y_p = y_g \mid s_0, s_1, \cdots, s_T, x\right)\right] \tag{5}$$

where $P\left(y_p = y_g \mid s_0, s_1, \cdots, s_T, x\right)$ represents the probability of obtaining an accurate final answer given the problem x and reasoning trajectory $traj$.

Table 4: Notation Table

Character	Meaning
π_θ	policy LLM
τ_D	task
D	demonstration examples of τ_D in in-context learning
x	input question / problem
y_p	predicted / decoded answer
y_g	gold standard answer
$traj$	trajectory / solution
T	number of reasoning steps
s_t	t-th reasoning step of trajectory $traj$
S_t	t-th state, which consists of input x and preceding reasoning steps $(s_1, s_2, ..., s_{t-1})$
a_t	t-th action based on the previous state S_{t-1}
s	node s in the tree structure
p	parent node of s
$Q(s)$	reward value of node s
p_φ	process reward model
o_ψ	outcome reward model
D_s	seed data
D_t	test data

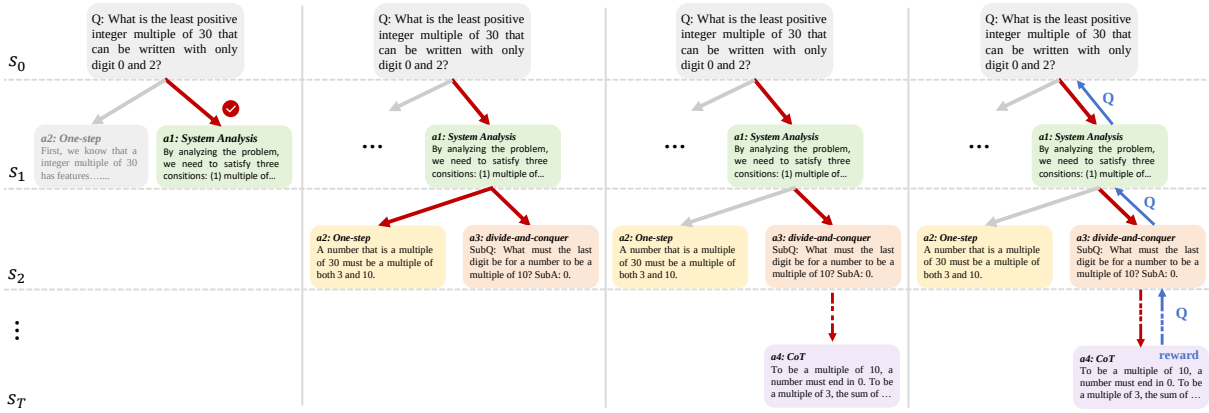
A.3 In-Context Learning

Originally introduced by (Brown et al., 2020), in-context learning (ICL) can be described as "*A process where the model is provided with natural language instructions and/or a few examples of a task, and is then expected to complete additional instances of that task simply by predicting what should come next.*" For a task τ_D, it typically involves two key components: a demonstration example space D and a joint probability distribution $P(X, Y)$ (Zhou et al., 2024c). The task demonstration $D = (x_i, y_i)_{i=1}^n$ contains n example pairs sampled from the joint distribution. These pairs typically consist of a problem x and the corresponding solution trajectory $traj$. Therefore, the ICL-based reasoning process can be formally expressed as:

$$\begin{aligned} D &\sim P(X, Y), \\ y_p &= \pi_\theta(D, x) \end{aligned} \tag{6}$$

Extensive research has focused on constructing high-quality examples and enriching the demonstration space D (Luo et al., 2024; Zhou et al., 2024c). For example, Chain-of-Thought (CoT) reasoning (Wei et al., 2022; Kojima et al., 2022) incorporates prompts like "Let's think step by step" alongside step-by-step reasoning examples, allowing models to emulate human-like reasoning and achieve success in complex problem-solving (Sprague et al., 2024). Self-Consistency (Wang et al., 2023b) further enhances performance by generating multiple reasoning paths and selecting the most consistent answer.

However, this traditional ICL paradigm primarily emphasizes example-level analogical learning, with performance constrained by the selection of demonstrations (Lu et al., 2023). This typically necessitates human expert intervention for complex reasoning tasks (Wang et al., 2023a; Zhao et al., 2024). In contrast, our approach shifts the focus from specific examples to high-level reasoning patterns, expanding the concept of context. This shift improves generalization, enabling fully automated, efficient inference without human intervention, even for models with fewer than 10 billion parameters. The most relevant prior work is Buffer-of-Thought (BoT) (Yang et al., 2024c), which designs thought templates for tasks by retrieving relevant templates to prompt LLMs. However, BoT requires human experts to predefine abstract thought templates and heavily relies on expensive high-performance models (e.g., GPT-4 in the original paper) for buffer augmentation. In contrast, our method is a fully automated reasoning process in which only small models are involved throughout the entire procedure, thus offering enhanced generalization and efficiency.



(a) Selection (b) Expansion (c) Simulation (d) Backpropagation

Figure 6: An illustration of the four phases in an iteration of MCTS for complex reasoning tasks (A.4).

A.4 Monte Carlo Tree Search

As a heuristic search algorithm, MCTS has demonstrated remarkable success in complex reasoning and decision-making environments (Chaslot et al., 2008; Ye et al., 2021; Zhou et al., 2024b,a). The algorithm conceptualizes search spaces as tree structures and has achieved significant breakthroughs across various domains, most notably in game-playing AI such as AlphaGo and AlphaZero (Dong et al., 2020). The basic MCTS algorithm involves an iterative search process with four key steps: selection, expansion, simulation, and backpropagation. As an example in mathematical reasoning, Figure 6 illustrates the four phases in an iteration, expanding the tree and then updating reward values.

Leveraging MCTS, recent approaches like AlphaMath (Chen et al., 2024) and rStar (Qi et al., 2024) exploit LLMs' intrinsic capabilities for iterative exploration to enhance complex reasoning. However, these methods employ a uniform search process across problems of varying difficulty and types, which often results in time-intensive computations. In contrast, our approach employs MCTS only during the generation of prior reasoning patterns (referred to as "thought cards" in Sec. 3.2) and references these thought cards during inference to achieve efficient reasoning. This design enables HiAR-ICL to adaptively match reasoning strategies to the complexity of each problem, significantly reducing the time complexity compared to traditional tree search methods. Additionally, it maintains a comprehensive search space and overall performance, thus achieving an optimal trade-off between efficiency and effectiveness.

A.5 Verification Methods

Selecting the most accurate reasoning trajectory among multiple candidate solutions presents a critical challenge. This paper explores three approaches: Self-Consistency (SC), Outcome Reward Model (ORM), and Process Reward Model (PRM).

SC Inspired by the intuition that a complex reasoning problem typically admits multiple different ways of thinking leading to its unique correct answer, we consider using a simple yet effective method, self-consistency (Wang et al., 2023b). For a problem p and its corresponding multiple reasoning trajectories $TRAJ = (traj_i, y_i)_{i=1}^{n}$, we perform voting to find the most consistent answer in the final answer set:

$$traj_{rm} = \arg\max_{traj} \sum_{i=1}^{N} \mathbb{I}\left(y_i = y\right) \quad (7)$$

ORM Given a problem p and its reasoning trajectory $traj$, ORM ($\mathcal{D} \times \mathcal{S} \rightarrow \mathbb{R}$) assigns a single real-value to indicate whether $traj$ is correct. ORM is usually trained with a cross-entropy loss (Cobbe et al., 2021; Li et al., 2023):

$$\mathcal{L}_{\text{ORM}} = y_g \log r_{traj} + (1 - y_g) \log(1 - r_{traj}) \quad (8)$$

where y_g is the golden answer ($y_g = 1$ if $traj$ is correct else $y_g = 0$), and r_{traj} is the sigmoid score of $traj$ assigned by ORM. The effectiveness of ORM is highly correlated with the quality of the constructed training dataset. Since reasoning problems typically have certain answers, researchers can automatically construct the ORM training set by: 1) Sampling candidate solutions 2) Labeling

Table 5: Comparison with other tree-based search methods. Note that, most method contain limited space. In contrast, we define a rich set of reasoning actions, thus enhancing the upper bound of model reasoning capabilities.

Method	Action Space
Tree-of-Thought (Yao et al., 2023) AlphaMath (Chen et al., 2024) AlphaLLM (Tian et al., 2024) MindStar (Kang et al., 2024) ReST-MCTS* (Zhang et al., 2024a)	a_2: one-step thought
RAP (Hao et al., 2023)	a_4: divide and conquer
BEATS (Sun et al., 2024)	a_1: system analysis, a_2: one-step thought, a_5: self-refinement
MCTSr (Zhang et al., 2024b) LLaMA-Berry (Zhang et al., 2024c)	a_3: chain-of-thought, a_5: self refinement
Ours	a_1, a_2, a_3, a_4, a_5

each solution based on whether it reaches the correct answer. Although false positives (incorrect reasoning to the right answer) will be mislabeled, prior work has shown this approach remains effective for training good ORM (Lightman et al., 2024; Wang et al., 2024c).

PRM To address ORM's difficulty in identifying false positives, PRM ($\mathcal{D} \times \mathcal{S} \rightarrow \mathbb{R}^+$) aims to score each step s_t in the reasoning trajectory (Li et al., 2023):

$$\mathcal{L}_{\text{PRM}} = \sum_{t=1}^{T} y_{s_t} \log(r_{s_t}) + (1 - y_{s_t}) \log(1 - r_{s_i}) \tag{9}$$

where y_{s_t} is the golden answer ($y_{s_t} = 1$ if t-th step s_t is correct else $y_{s_t} = 0$), and r_{s_t} is the PRM output sigmoid score. Specifically, Lightman et al. (2024) frames PRM training as a three-class classification task, where human annotations categorize steps as 'good', 'neutral', or 'bad'. Similarly, MATH-SHEPHERD (Wang et al., 2024c) collects random rollout trajectories through the Best-of-N policy and combines process rewards to create the PRM training dataset automatically. ReST-MCTS* (Zhang et al., 2024a) integrates process reward guidance with tree search to explore efficient solution spaces and synthesize high-quality trajectories. With a trained PRM, we can score each step in the reasoning trajectory for a test question q_t. Following prior work (Li et al., 2023; Wang et al., 2024c), we utilize the minimum score across all steps to represent the final score of a solution s:

$$traj_{rm} = \arg\max_{traj} \sum_{i=1}^{N} \mathbb{I}\left(y_i = y\right) \cdot rm\left(p, traj_i\right) \tag{10}$$

where $rm\left(p, traj_i\right)$ is the score of i-th solution assigned by reward model. Based on the scores of each candidate solution, we select the trajectory with the highest final score as the ultimate solution for q_t.

B Algorithm Details

Overview The pseudocode for HiAR-ICL is shown in Algorithm 1, which consists of four components: 1) define atomic reasoning actions 2) construct thought cards via MCTS 3) select reasoning patterns 4) solve and verify.

B.1 Action Space

Emerging research suggests that the upper bound of model reasoning capabilities is closely correlated with the available action space (Wu et al., 2024). As illustrated in (Qi et al., 2024), existing approaches typically involve a restricted action space (Table 5), which may hinder LLM's full reasoning potential. Therefore, we propose a more expansive framework, which contains five atomic reasoning actions as follows:

- (a_1) *System Analysis (SA)*: Analyzing the overall structure of the problem and identifying the constraints and conditions before addressing it, thereby clarifying task requirements effectively.

Algorithm 1 The proposed high-level automated reasoning paradigm HiAR-ICL

Input: language mode π_θ, seed data D_s, test data D_t, number of rollouts K, branch b, depth limit d_{max}, exploration weight w, reward update parameter α, VOC balance factor k, verIfication function $f_v(\cdot)$
Output: the optimal reasoning trajectory For each test problem in D_t $traj_{best}$

Initialize action space $A = \{a_1, a_2, a_3, a_4, a_5\}$ ▷ **Sec. 3.1 define atom reasoning actions**
Initialize node reward value function $Q(s) : S \mapsto \mathbb{R}$ and visit counter $N : S \mapsto \mathbb{N}$ to zero
$Cards \leftarrow \{\}$ ▷ **Sec. 3.2 construct thought cards via MCTS**

——— *Sec. 3.2 step 1: acquire reasoning paths For seed data* ———
for data point (q, y_q) in D_s **do**
  $QP_{list} \leftarrow [\,]$
  $T_q \leftarrow$ Initialize_tree(q)

  for $i = 1$ in range(K) **do**
    $C \leftarrow$ root(T_q) *// MCTS phase 1: selection*
    if C has child nodes **then**
      $C \leftarrow \text{argmax}_{C' \in \text{children}(C)} \left(\frac{Q(C')}{N(C')} + w\sqrt{\frac{\ln N(C)}{N(C')}} \right)$
    end if

    $\mathbb{C}' \leftarrow$ Get_child_nodes(C) *// MCTS phase 2: expansion*
    for j in range(b) **do**
      $\mathbb{C}' \leftarrow \mathbb{C}' \cup$ Get_new_child(C)
    end for

    while C is not leaf node or node depth $< d_{max}$ **do**
      $\mathbb{C}' \leftarrow$ Get_child_nodes(C) *// MCTS phase 3: simulation*
      $C \leftarrow$ Random_sample($\mathbb{C}'$)
    end while

    $T \leftarrow$ the actual depth of final visited node
    if s_T is leaf node and reaches correct answer y_q **then**
      $Q(s_T) \leftarrow 1$
    end if

    for t in range($T-1$, 0, -1) **do**
      $N(s_t) \leftarrow N(s_t) + 1$ *// MCTS phase 4: backpropagation*
      $Q(s_t) \leftarrow (1-\alpha)Q(s_t) + \alpha(s_{t+1})$ *// update value and visit count of the rollout node*
    end for

  end for

  $\mathbb{P} \leftarrow$ Obtain_valid_paths(T_q)
  $p_{best} \leftarrow \text{argmax}_{p_q \in \mathbb{P}} k \cdot Reward(p_q) + (1-k) \cdot Cost(p_q)$
  QP_{list}.append($[q, p_{best}]$) *// construct candidate question-path repository*

end for

——— *Sec. 3.2 step 2: distill paths into thought cards* ———
$Store \leftarrow$ Reasoning_pattern_aggregation(QP_{list})
for rp, q_{rp} in $Store$ **do**
  $cc \leftarrow$ Obtain_cognitive_complexity(q_{rp})
  $Cards$.update($\{rp : cc\}$)
end for

for data point (q_t, y_p) in D_t **do**
  $SC =$ pattern_match(q_t, $Cards$) ▷ **Sec. 3.3 select reasoning patterns**
  $score_{dict} \leftarrow \{\}$

  for selected card c in SC **do**
    $traj = \pi_\theta(q_t, c)$ ▷ **Sec. 3.4 solve and verIfy**
    $score_{dict}.update(\{traj : f_v(q_t, traj)\})$
  end for

  $traj_{best} \leftarrow \text{argmax}_{traj \in \text{Traj}} score_{dict}[traj]$ *// optimal solution through HiAR-ICL*

end for

- (a_2) *One-Step Thought (OST)*: Generating the next one-step thought based on the given question and the preceding reasoning steps.
- (a_3) *Chain-of-Thought (CoT)*: Facilitating step-by-step reasoning by constructing a logical sequence of intermediate thoughts, where each step incrementally builds on the previous ones.
- (a_4) *Divide and Conquer (DC)*: Breaking down a complex reasoning problem into several smaller subproblems and progressively solving them to achieve the overall solution.
- (a_5) *Self-Reflection and Refinement (SRR)*: Engaging in timely reflection of prior solutions and implementing necessary refinement during the reasoning process to ensure accuracy.

B.2 Reward Value in MCTS

For simplicity, we avoid introducing an additional reward model for scoring. Given the current skepticism regarding the self-rewarding capabilities of LLMs, alternative methods are necessary. Inspired by the principle that actions leading to correct answers should be rewarded more frequently, we aim to increase their likelihood of selection in future MCTS tree expansions. Following (Qi et al., 2024), we define the reward value as the likelihood (or confidence) of self-consistency via majority voting. Note that this principle applies only to leaf nodes. The Q-values of intermediate nodes are initially set to 0 and are dynamically updated during the backpropagation process, as described in 1.

B.3 Thought Cards Selection

Within our proposed cognitive complexity framework, we define three metrics: (1) Subquestion Count (SC): quantifying the number of decomposable subproblems; (2) Problem Condition Complexity (PCC): measuring the number of distinctive problem conditions; and (3) Semantic Similarity (SS): assessing the semantic distance between the target problem and the seed dataset. As presented in Table 3 in *Section 5.4*, our empirical analysis reveals minimal statistical variations among these metrics. Consequently, we adopt PCC as the primary metric. We conjecture that future research could productively explore sophisticated integration strategies, such as weighted ranking summation, to comprehensively leverage these three metrics and develop a more nuanced quantification of reasoning problem complexity.

B.4 Verification

For verification, we utilize off-the-shelf pre-trained models: Llama3.1-8B-ORM-Mistral-Data[1] (ORM) and Llama3.1-8B-PRM-Mistral-Data[2] (PRM). By default, our experimental results are reported based on PRM verification. Moreover, as demonstrated in Table 3 in *Section 5.4*, our method still achieves notable performance even in scenarios lacking a readily available verification model, such as when relying solely on the simple yet effective method, self-consistency. In future work, we aim to explore more sophisticated verification approaches to more precisely select the optimal reasoning trajectory.

C Experimental Settings

C.1 Models

HiAR-ICL is a general approach applicable to various LLMs. In our experiments, we evaluate its effectiveness using powerful open-source models: Llama3-8B-Instruct (Dubey et al., 2024), Llama3.1-8B-Instruct (Meta AI, 2024), Yi-1.5-6B-Chat (Young et al., 2024), Qwen2-7B-Instruct (Yang et al., 2024a), and Qwen2.5-7B/14B-Instruct (Qwen Team, 2024). Unless stated otherwise, all models are the Instruct version. In D.2, we also analyze the performance of HiAR-ICL combined with base models. By focusing on LLMs with parameter counts generally under 10B, we aim to demonstrate the robustness and efficiency of our method. We expect that applying HiAR-ICL to small language models will achieve results comparable to or exceeding closed-source LLMs.

C.2 Datasets

We conduct main experiments on four popular multi-hop reasoning datasets. We briefly describe and list them in Table 6.

- **GSM8K (Cobbe et al., 2021)**: This dataset contains 7,473 training and 1,319 testing grade-school math word problems, which require between 2 to 8 steps for resolution. The solutions primarily involve performing a sequence of basic arithmetic operations (addition, subtraction, multiplication, and division) to arrive at the final answer.
- **SVAMP (Patel et al., 2021)**: Similar to GSM8K, this dataset contains a 1,000-sample

[1] RLHFlow/Llama3.1-8B-ORM-Mistral-Data
[2] RLHFlow/Llama3.1-8B-PRM-Mistral-Data

Table 6: Detailed information on the datasets utilized in the main result. Seed Datasets are utilized to construct thought cards, and test sets for final evaluation.

Category	Dataset	#Seed Samples	#Test Samples
Arithmetic	GSM8K (Cobbe et al., 2021)	200	1319
	SVAMP (Patel et al., 2021)	200	300
Mathematics	MATH (Hendrycks et al., 2021)	200	500
Commonsense	StrategyQA (Geva et al., 2021)	200	687

Table 7: All hyperparameters utilized in this paper.

Hyperparameter	Value	Description
temperature	0.8	vllm inference settings
top-p	0.9	
top-k	40	
repetition_penalty	1.0	
max_new_tokens	1024	
maximum tree depth d_{max}	5	MCTS
exploration weight w	2.0	
predefined terminal threshold c	0.90	
balance factor k	0.95	VOC-based optimal path selection in *Sec. 3.2* Step 2

testing set of elementary-level mathematical word problems (MWPs). These problems exhibit diverse structural variations and challenge large language models (LLMs) to generate precise numerical values or equations for solution. Following the existing dataset partition[3], we extract 200 samples from the 700-sample training set as the seed dataset and utilize the remaining 300-sample test set for evaluation.

- **MATH (Hendrycks et al., 2021)**: This dataset includes 12,500 challenging competition-level mathematics problems. Each problem is accompanied by a complete step-by-step solution, which can be leveraged to teach models to generate answer derivations and explanations. Following prior work (Qi et al., 2024; Zhang et al., 2024a), we utilize MATH500 as our test set for a fair comparison, a representative and highly challenging 10% subset of MATH.

- **StrategyQA (Geva et al., 2021)**: This dataset comprises 2,780 examples, each consisting of a strategy question, its decomposition, and supporting evidence paragraphs. We utilize the 687 examples from the test set for evaluation.

[3]https://huggingface.co/datasets/ChilleD/SVAMP

For the seed dataset D_s, we randomly sample 200 instances from the training sets of each dataset to construct thought cards. The sampling's randomness ensures algorithmic robustness. Additionally, only 200 samples are used for tree search, and cross-distribution transfer capabilities have been validated in the main text. Therefore, in practical scenarios, publicly available datasets can be utilized to construct the seed dataset, enabling the construction of thought cards and final reasoning. This ensures the generality and user-friendliness of the approach.

C.3 Baselines

For baselines compared in this paper, we run ICL approaches such as CoT and self-consistency to record their performance. For other tree-based methods and closed-source models, we directly reference their original reported results.

C.4 Evaluation Details

As detailed in the main paper, we evaluate our approach using two primary metrics. First, we report accuracy as our primary evaluation metric, where correctness is determined by comparing the model's final answer with the ground truth. To ensure consistent answer extraction, we require the

Table 8: Comparison with leading LLMs. The best results in each box are highlighted in **bold**. Results for off-the-shelf models are sourced from corresponding official websites. Note that, Qwen2.5-7B-Instruct model outperforms larger models and even powerful closed-source models.

MODEL	Size	MATH	GSM8K
Closed-Source Models			
GPT-4o-0806 (OpenAI, 2024)	-	**76.6**	96.1
Claude-3.5-Sonnet-0620 (Anthropic, 2024a)	-	71.1	**96.4**
GPT-4o mini-0718 (OpenAI, 2024)	-	70.2	93.2
Gemini-1.5-Pro (Google DeepMind, 2024)	-	67.7	90.8
Claude-3-Opus-0304 (Anthropic, 2024b)	-	60.1	95.0
GPT-4 (OpenAI, 2023)	-	64.5	94.2
GPT-3.5 (OpenAI, 2022)	-	43.1	81.6
Open-Source Models (>30B)			
Llama3.1-405B-Instruct (Meta AI, 2024)	405B	73.8	**96.8**
Nemotron4-340B-Instruct (Adler et al., 2024)	340B	41.1	92.3
DeepSeek-v2.5 (Liu et al., 2024)	236B	74.7	95.1
Mixtral-8x22B-Instruct (Mistral AI team, 2024a)	141B	54.1	88.2
Mixtral-large2-Instruct (Mistral AI team, 2024b)	123B	69.9	92.7
Qwen2-72B-Instruct (Yang et al., 2024a)	72B	69.0	93.2
NuminaMath-72B CoT (Beeching et al., 2024)	72B	66.7	90.8
Llama3.3-70B-Instruct (Meta AI, 2024)	70B	**77.0**	-
Llama3.1-70B-Instruct (Meta AI, 2024)	70B	68.0	95.1
Llama3-70B-Instruct (Dubey et al., 2024)	70B	50.4	93.0
Yi-1.5-34B-Chat (Young et al., 2024)	34B	50.1	90.2
Ours (≤14B)			
Qwen2.5-14B-instruct (Qwen Team, 2024)	14B	**81.4**	**95.8**
Qwen2.5-7B-instruct (Qwen Team, 2024)	7B	80.6	93.7
Qwen2-7B-instruct (Yang et al., 2024a)	7B	66.8	91.8
Yi-1.5-6B-Chat (Young et al., 2024)	6B	57.4	86.4
Llama3-8B-Instruct (Dubey et al., 2024)	8B	46.6	89.6
Llama3.1-8B-Instruct (Meta AI, 2024)	8B	58.0	90.7

LLM to explicitly state its solution following a predefined format (e.g., "The answer is"). Second, we measure the average reasoning time to assess our method's computational efficiency relative to existing search-based approaches. All time measurements were recorded on the same device (A100 GPUs), thereby ensuring fair comparisons.

C.5 Implementation Details

We use the vLLM framework[4] with parameters: temperature 0.8, top p 0.9, and max tokens 1024. All experiments are conducted on a Ubuntu 22.04 machine equipped with NVIDIA A100-80GB GPUs. We list hyperparameters in Table 7.

[4] https://github.com/vllm-project/vllm

D Additional Experiments

This section presents comprehensive results and analysis, including: comparison with powerful LLMs (D.1), performance across problem complexity (D.3), integration with supervised fine-tuning (D.2), and ablation studies (D.4, effect of atomic action and test-time scaling law).

D.1 Comparison with Powerful LLMs

Table 8 provides a performance comparison of our method against leading open-source and closed-source models. By employing HiAR-ICL, our 14B and 7B Qwen2.5 models outperform numerous state-of-the-art closed and open-source models, achieving superior performance.

Table 9: Ablation results on Llama3-8B-Instruct. HiAR-ICL requires every component for optimal performance. △ indicates the performance gain of HiAR-ICL compared to the best baseline.

Model	Setting	MATH	GSM8K	SVAMP	StrategyQA	Average	△ (↑)
Llama3-8B	Zero-shot CoT	5.8	17.7	30.0	9.6	15.8	-
	Few-shot CoT	13.2	39.4	56.7	61.1	42.6	-
	HiAR-ICL (Ours)	**34.4**	**81.4**	**90.0**	**70.7**	**69.1**	+26.5
Llama3-8B-Instruct	Zero-shot CoT	5.8	68.3	70.9	57.2	50.5	-
	Few-shot CoT	17.8	74.5	81.0	68.4	60.4	-
	HiAR-ICL (Ours)	**46.6**	**89.6**	**92.7**	**73.4**	**75.6**	+15.2
Llama3.1-8B	Zero-shot CoT	7.6	18.4	29.0	5.7	15.2	-
	Few-shot CoT	14.8	40.4	59.0	61.3	43.9	-
	HiAR-ICL (Ours)	**38.4**	**82.3**	**90.0**	**71.1**	**70.5**	+26.6
Llama3.1-8B-Instruct	Zero-shot CoT	18.0	61.5	69.3	52.4	50.3	-
	Few-shot CoT	47.2	76.6	82.0	63.6	67.3	-
	HiAR-ICL (Ours)	**58.0**	**90.7**	**93.0**	**75.7**	**79.4**	+12.1

Table 10: Performance variations of Llama3-8B-Instruct and Qwen2.5-7B-Instruct across different difficulty levels on MATH. We list the result of Zero-shot CoT, few-shot CoT+SC, and our method.

Method	1	2	3	4	5	Average
Llama3-8B-Instruct						
CoT	53.5	34.3	22.1	8.2	3.7	17.8
CoT+SC	62.5	52.6	38.1	19.5	9.7	28.8
HiAR-ICL	88.4	75.6	56.2	32.0	16.5	46.6
Qwen2.5-7B-Instruct						
CoT	95.3	87.7	79.0	67.2	40.3	68.6
CoT+SC	95.3	90.0	91.4	73.4	52.2	76.4
HiAR-ICL	97.7	94.5	92.3	81.1	53.7	80.6

Table 11: Effectiveness of each atomic action in our proposed action space. We evaluate on Llama3-8B-Instruct.

Action Space	MATH	StrategyQA
a_3 (i.e. CoT)	17.8	68.4
$a_3 + a_4$	42.6	69.9
$a_2 + a_3 + a_4$	43.2	70.3
$a_1 + a_2 + a_3 + a_4$	46.0	72.6
All ($a_1 + a_2 + a_3 + a_4 + a_4$)	46.6	73.4

D.2 Integration with SFT

Our method can be effectively combined with Supervised Fine-Tuning (SFT). As shown in Table 9, our approach achieves promising results on the base model. Moreover, when applied to instruction-tuned models, the performance is further enhanced, demonstrating the potential synergy between our method and SFT. In future work, we aim to explore more sophisticated integration strategies, such as incorporating SFT processes into HiAR-ICL to enable efficient card construction and reasoning pattern selection.

D.3 Performance across Problem Complexity

As shown in Table 10, we present the performance of Llama3-8B-Instruct and Qwen2.5-7B-Instruct on Zero-shot CoT, Few-shot CoT+SC, and our method, HiAR-ICL, across different difficulty levels of the challenging MATH dataset. Taking Qwen2.5-7B-Instruct as an example, our approach improves performance across all levels, with an average accuracy boost of 2.6% for the easier levels 1-3. Notably, for the more difficult level 4, the improvement reaches +7.7%. This indicates that our method has the potential to solve more challenging problems and enhance reasoning performance. This may be due to the introduction of high-level reasoning patterns, which help LLMs find a clearer solution more quickly.

D.4 Ablation Studies

In addition to the ablation studies on modules 2 (thought cards construction), 3 (reasoning pattern selection), and 4 (verification) in the main text, we present additional results on module 1 (*effect of each atomic action*) and investigate whether HiAR-ICL satisfies the *test-time scaling law*. These supplementary analyses aim to provide a more comprehensive understanding of the functional roles of each module in HiAR-ICL.

Effect of Each Atomic Action We conduct an ablation study to evaluate the effectiveness of each atomic action within our proposed action space

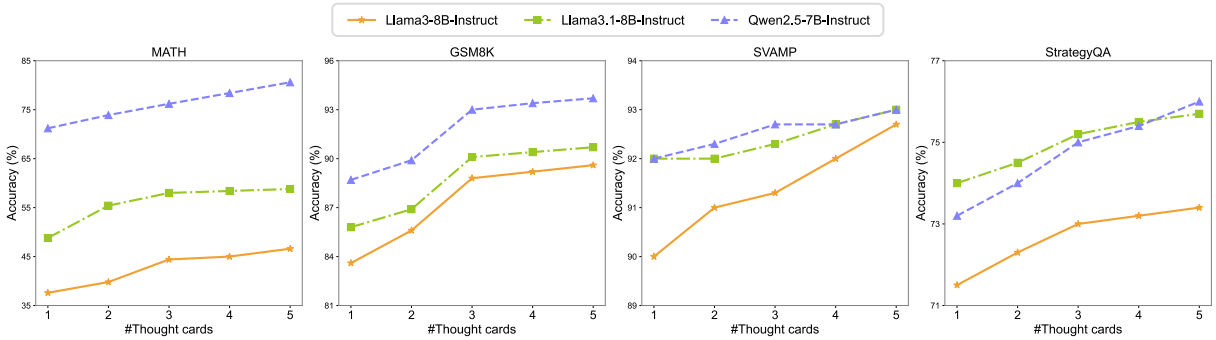


Figure 7: Test-time scaling law performance. We examine the variation in HiAR-ICL performance with the number of selected reasoning guidance-providing thought cards in Section 3.3.

using the Llama3-8B-Instruct model. Our experiments in Table 11 highlight the critical importance of individual action components. Compared to existing approaches with typically restrictive action spaces (as shown in Table 5), our expanded action space yields significant performance improvements. The study may open avenues for future research, particularly in developing adaptive action space selection mechanisms for broader problem domains. We aim to enhance approach generalization by dynamically customizing action spaces, with a specific focus on emerging multimodal reasoning tasks.

Test-time Scaling Law in Reasoning Tasks To investigate whether our method follows test-time scaling laws, we gradually increase the number of selected thought cards during inference. As shown in Figure 7, progressively adding more reasoning guidance-providing thought cards consistently improves performance for our HiARD-ICL approach. However, it is important to note that this increase in thought cards also brings about additional computational overhead. Therefore, finding an optimal balance between performance and cost becomes a key direction for future research and optimization.

E Prompts

We reference and adapt prompts designed by other researchers [5] to construct the input prompts for our five actions in Figure 8-12. To improve readability, we have applied selective formatting, such as bold highlighting, to enhance the presentation.

[5] https://github.com/reasoning-machines/prompt-lib/tree/main and https://github.com/zhentingqi/rStar/tree/main/prompts

F Case Studies

Figures 13-17 present detailed problem-solving examples employing our method and in-context learning. Our approach demonstrates notable advantages in addressing relatively challenging problems, effectively leveraging the rich action space and guidance-providing thought cards to achieve precise reasoning. For instance, in Figure 13, both ICL and GPT-4 (results) output incorrect results, while our method with Qwen2.5-7B-Instruct provides a logically clear solution process and ultimately arrives at the correct answer.

Action 1. System Analysis (SA)

<Instruction>
You are an AI assistant to help me analyzing the overall structure of the problem and identifying the constraints or conditions before addressing it. You need to list all constrains as conditions, remember to fully express the information in the original question.

<Action Description>
Q: Genghis Khan: founder and first Great Khan of the Mongol Empire. Are more people today related to Genghis Khan than Julius Caesar?
A: Given a list of conditions, please answer the question. Condition 1: Genghis Khan was the founder and first Great Khan of the Mongol Empire. Condition 2: Julius Caesar was a Roman general and statesman. Question: Are more people alive today related to Genghis Khan than to Julius Caesar?

Q: Angelo and Melanie want to plan how many hours over the next week they should study together for their test next week. They have 2 chapters of their textbook to study and 4 worksheets to memorize. They figure out that they should dedicate 3 hours to each chapter of their textbook and 1.5 hours for each worksheet. If they plan to study no more than 4 hours each day, how many days should they plan to study total over the next week if they take a 10-minute break every hour, include 3 10-minute snack breaks each day, and 30 minutes for lunch each day?
A: Given a list of conditions, please answer the question. Condition 1: Angelo and Melanie need to study 2 textbook chapters and 4 worksheets. Condition 2: They allocate 3 hours per textbook chapter and 1.5 hours per worksheet. Condition 3: Their daily study limit is 4 hours, with a 10-minute break every hour, three 10-minute snack breaks, and a 30-minute lunch break each day. Question: Over the next week, for how many days should they plan to study to cover all their materials?
...

<User Input>
Q: given question

Figure 8: Input prompt for action 1: system analysis.

Action 2. One-Step Thought (OST)

<Instruction>
A chat between a curious user and an AI assistant. The assistant gives one-step thought to the user's questions. At the final step, a conclusive answer is given in the format of "The answer is: <ANSWER>.", where <ANSWER> should be a numeric answer, math expression, or bool value ("true" or "false").

<Action Description>
Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?
A: Let's think step by step.
Step 1: Identify the initial number of trees. The problem states there are 15 trees in the grove. ***(Existing steps)***
Step 2: Identify the final number of trees. The problem states that there will be 21 trees after the workers are done planting. ***(one-step thought output)***

Q: Leah had 32 chocolates and her sister had 42. If they ate 35, how many pieces do they have left in total?
A: Let's think step by step.
Step 1: Add the number of chocolates Leah and her sister had initially. Leah had 32 chocolates and her sister had 42 chocolates. So, they had 32 + 42 = 74 chocolates in total. ***(one-step thought output)***
...

<User Input>
Q: given question

Figure 9: Input prompt for action 2: one-step thought.

Action 3. Chain-of-Thought (CoT)

<Instruction>
A chat between a curious user and an AI assistant. The assistant gives step-by-step solutions to the user's questions. At the final step, a final answer is given in the format of \"The answer is: <ANSWER>.\", where <ANSWER> should be a numeric answer, math expression, or bool value ("true" or "false").

<Action Description>
Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?
A: Let's think step by step. There are 15 trees originally. Then there were 21 trees after some more were planted. So there must have been 21 - 15 = 6. The answer is: 6.
...

<User Input>
Q: given question

Figure 10: Input prompt for action 3: chain-of-thought.

Action 4. Divide and Conquer (DC)

<Instruction>
Given a question, please decompose it into sub-questions. For each sub-question, please answer it in a complete sentence, ending with "The answer is <a numeric answer, math expression, or bool value (true or false).>". When the original question is answerable, please start the subquestion with "Now we can answer the question: <original question>".

<Action Description>
Q: Genghis Khan: founder and first Great Khan of the Mongol Empire. Are more people today related to Genghis Khan than Julius Caesar?Question 1: Four years ago, Kody was only half as old as Mohamed. If Mohamed is currently twice as 30 years old, how old is Kody?
A: *SubQ 1.1:* How old is Mohamed currently?
SubA 1.1: Mohamed is twice as old as 30 years, which means he is 30 * 2 = 60 years old.
SubQ 1.2: What was Kody's age four years ago, given that it was half of Mohamed's age at that time?
SubA 1.2: Four years ago, Mohamed was 60 - 4 = 56 years old, so Kody was half of that, which is 56 / 2 = 28 years old.
SubQ 1.3: Now we can answer the question: How old is Kody?
SubA 1.3: Kody is currently 28 + 4 = 32 years old. The answer is 32.
...

<User Input>
Q: given question

Figure 11: Input prompt for action 4: divide and conquer.

Action 5. Self-Reflection and Refinement (SRR)

<Instruction>
Given a question, partial solution steps, and potential errors, analyze the solution steps and correct any mistakes.

Guidlines:
1. Evaluate each solution step provided, identifying any errors in logic.
2. For each identified error, explain the reasoning in the <THINKING> section.
3. Provide a corrected solution in the <CORRECT> section.
4. Ensure the final output maintains the specified format.

Purpose:
Help clarify and correct reasoning step-by-step, displaying critical thinking and clear solution paths. Note: The provided solution is generally incomplete; do not attempt to fill in missing parts. Focus solely on analyzing the given solution. Maintain the original language style, including formatting and notation of formulas.

Output Format:
<THINKING>: Use this section to think through and identify any mistakes.
<CORRECT>: Provide the corrected solution in this section.

<Action Description>
Question:
Leah had 32 chocolates and her sister had 42. If they ate 35, how many pieces do they have left in total?
Some Steps:
Step 1: Add the number of chocolates Leah and her sister had initially. Leah had 32 chocolates and her sister had 42 chocolates. So, they had 32 + 42 = 73 chocolates in total.
Step 2: Subtract the number of chocolates they ate from the total number of chocolates they had. They ate 35 chocolates. So, they have 73 - 35 = 38 chocolates left.

<THINKING>:
Step 1: This step is incorrect. Adding the number of chocolates Leah and her sister had initially results in 32 + 42 = 74, not 73.
Step 2: This step is incorrect. Since the total was 74 chocolates and they ate 35, the calculation should be 74 - 35 = 39, not 38.

<CORRECT>:
Step 1: Add the number of chocolates Leah and her sister had initially. Leah had 32 chocolates and her sister had 42 chocolates. So, they had 32 + 42 = 74 chocolates in total.
Step 2: Subtract the number of chocolates they ate from the total number of chocolates they had. They ate 35 chocolates. So, they have 74 - 35 = 39 chocolates left.
...

<User Input>
Q: given question

Figure 12: Input prompt for action 5: self-reflection and refinement.

Example 1 on MATH – Number Theory

Question What is the least positive integer multiple of 30 that can be written with only the digits 0 and 2?

Ground Truth

Let M be the least positive multiple of 30 that can be written with only the digits 0 and 2.
First, M is a multiple of 10, so its units digit must be 0.
M is also a multiple of 3, which means the sum of its digits must be a multiple of 3.

Therefore, we must take at least three 2's. Since M is minimal, we take exactly three 2's and do not have any additional 0's: $M=\\boxed{2220}$.

2220

GPT-4o response: *https://chatgpt.com/share/675d6e50-683c-8012-b12d-f07c6ad41031*

We are tasked with finding the least positive integer multiple of 30 that can be written using only the digits 0 and 2. A number that is a multiple of 30 must satisfy two conditions:
1. It must be divisible by 2. 2. It must be divisible by 3.

Divisibility by 2:
For divisibility by 2, the last digit of the number must be even. Since the only digits allowed are 0 and 2, the last digit must be 2.

Divisibility by 3:
For divisibility by 3, the sum of the digits of the number must be divisible by 3. Since the digits are restricted to 0 and 2, we need to ensure that the sum of the digits is divisible by 3.

Let's start by examining small numbers made up of the digits 0 and 2, checking each for divisibility by both 2 and 3.

Step-by-Step Process:

2:	a. Last digit is 2 (divisible by 2).	b. Sum of digits is 222, which is not divisible by 3.	c. Not divisible by 3.
20	a. Last digit is 0 (divisible by 2).	b. Sum of digits is 2+0=2, which is not divisible by 3.	c. Not divisible by 3.
22	a. Last digit is 2 (divisible by 2).	b. Sum of digits is 2+2=4, which is not divisible by 3.	c. Not divisible by 3.
202	a. Last digit is 2 (divisible by 2).	b. Sum of digits is 2+0+2=4, which is not divisible by 3.	c. Not divisible by 3.
220	a. Last digit is 2 (divisible by 2)	b. Sum of digits is 2+2+0=4, which is not divisible by 3.	c. Not divisible by 3.
222	a. Last digit is 2 (divisible by 2)	b. Sum of digits is 2+2+2=6, which is divisible by 3.	c. Divisible by both 2 and 3.

Thus, the least positive integer multiple of 30 that can be written with only the digits 0 and 2 is 222 ✗

Few-shot CoT response:

Let's think step by step.

1. We need a number that is divisible by both 2 and 3.

2. Since the number will only consist of the digits 0 and 2, it won't have any factors other than powers of 2 and 5.

3. Thus, we want to find the smallest such number that contains at least one factor of each power of 2 (to include all even numbers) and has a factor of 3 (so that it is divisible by 3).

4. Let $n = 2^{m} \\cdot 5^p$ for some non-negative integers m and p. The smallest possible value of n occurs when $m=1$, which means $n = 2 \\cdot 5^p$.

5. Since 5^p must contain at least two digits if it is greater than 1, we know that p cannot be less than 2. So the lowest possible value for n is when $p=2$, giving us $n = 2 \\cdot 25 = 50.$

6. However, this is not the correct solution because we also need our number to be divisible by 3, and 50 is not divisible by 3.

7. Instead, let's try adding another factor of 2, so that $n = 4 \\cdot 25 = 100,$ which is divisible by 3.

8. Therefore, $\\boxed{100}$ is the smallest such number.

The answer is: 100 ✗

HiAR-ICL response:

Step 1: Divide and Conquer
SubQ 1.1: What are the factors of 30?
SubA 1.1: The factors of 30 are 2, 3, and 5. The answer is 2, 3, and 5.

Step 2: Divide and Conquer
SubQ 1.2: What is the smallest digit that needs to appear at least once in our number to make sure it's divisible by 30?
SubA 1.2: To ensure divisibility by 30, which includes being divisible by 3 (sum of digits must be a multiple of 3), we need at least the digit 2 to satisfy this condition. The answer is 3.

Step 3: Chain-of-Thought
Let's think step by step.

1. To find the least positive integer multiple of 30 that can be written with only the digits 0 and 2, let's consider the factors of 30 as mentioned in Step 1. These factors are 2, 3, and 5.

2. From Step 2, we know that to be divisible by 30, the number must be divisible by both 3 and 5. For divisibility by 5, the last digit must be 0. For divisibility by 3, the sum of the digits must be a multiple of 3.

3. Starting with the smallest possible numbers composed of the digits 0 and 2:
a. **Two-digit numbers**:
20: Sum = 2 (not divisible by 3)
02: Not valid since leading zeros are not allowed
b. **Three-digit numbers**:
202: Sum = 2 + 0 + 2 = 4 (not divisible by 3)
220: Sum = 2 + 2 + 0 = 4 (not divisible by 3)
c. **Four-digit numbers**:
2020: Sum = 2 + 0 + 2 + 0 = 4 (not divisible by 3)
2200: Sum = 2 + 2 + 0 + 0 = 4 (not divisible by 3)
2220: Sum = 2 + 2 + 2 + 0 = 6 (divisible by 3) and ends in 0 (divisible by 5)

Since 2220 satisfies all conditions (divisibility by 2, 3, and 5), it is the smallest such number. Thus, the least positive integer multiple of 30 that can be written with only the digits 0 and 2 is \\(\\boxed{2220}\\).

The answer is: 2220 ✓

Figure 13: Example 1 about Number Theory on the MATH dataset. GPT-4o results are available at GPT-4o results

Example 2 on MATH – Precalculus

Question Suppose sin D = 0.7 in the diagram below. What is DE?
[asy]
pair D,E,F;\nF = (0,0);\nD = (sqrt(51),7);\nE = (0,7);\ndraw(D--E--F--D);
draw(rightanglemark(D,E,F,15));\nlabel(\"D\",D,NE);\nlabel(\"E\",E,NW);\nlabel(\"F\",F,SW);\nlabel(\"7\",(E+F)/2,W);
[/asy]

Ground Truth The triangle is a right triangle, so sin D = EF / DF. Then we have that sin D = 0.7 = 7 / DF, so DF = 10.
Using the Pythagorean Theorem, we find that the length of DE is \sqrt{DF^2 - EF^2}, or \sqrt{100 - 49} = \\boxed{\sqrt{51}}.

\sqrt{51}

GPT-4o response: *https://chatgpt.com/share/675d7a3e-ccd4-8012-89b1-71c1f5fd2954*

We are given a right triangle ΔDEF with a right angle at E, an we know that sinD=0.7. We need to find the length of DE.

Step 1: Identify given information
The triangle has the following properties:
1. ∠DEF=90 ^\\circ (right angle at E) 2. sin D = 0.7 3. The length of side EF=7

Let the sides of the triangle be:
1. DE (the side opposite to angle ∠DEF, which we need to find) 2. DF (the hypotenuse of the triangle)

Step 2: Use the sine definition
The sine of an angle in a right triangle is defined as the ratio of the length of the side opposite the angle to the length of the hypotenuse. For angle D, we have:
sin D=opposite / hypotenuse = EF / DF

From the problem, we know that sin D = 0.7 and EF=7, so we can write: 0.7 = 7 / DF

Step 3: Solve for DF
To solve for DF, multiply both sides of the equation by DF and then divide by 0.7: DF = 7/0.7 = 10
Thus, the length of the hypotenuse is 10

Step 4: Use the Pythagorean theorem
We now use the Pythagorean theorem to find the length of DE. The Pythagorean theorem states that in a right triangle: DE^2 + EF^2 = DF^2
Substitute the known values for EF=7 and DF = 10: DE^2 + 7^2 = 10^2, DE^2 + 49^2 = 100, DE^2 = 100-49=51

Step 5: Solve for DE
Take the square root of both sides to find DE: DE = \sqrt{51}

Thus, the length of DE is \sqrt{51}

Few-shot CoT response:

Let's think step by step.

1. By definition, sin D = EF / DF, so EF = DF·sin D = 7\\cdot0.7=4.9$.

2. Therefore, the length of segment DE is $\\boxed{4.9}$.

The answer is: 4.9

HiAR-ICL response:

Step 1: One-Step Thought
Identify the right triangle. Here, it's triangle DEF, where DE is perpendicular to EF, making angle DEF a right angle

Step 2: Chain-of-Thought
Let's think step by step.

1. From the problem, we know that ΔDEF is a right triangle with ∠DEF = 90^\\circ, and EF = 7.

2. Given that sin D = 0.7, we recognize that
sin D = opposite / hypotenuse = EF / DF = 7 / DF.

Thus, DF = 7 / sin D = 7 / 0.7 = 10.

3. Now, since ΔDEF is a right triangle, we can use the Pythagorean theorem to find DE:
DE^2 + EF^2 = DF^2

Substituting the known values, we get:
DE^2 + 7^2 = 10^2
DE^2 + 49 = 100
DE^2 = 51

Taking the square root of both sides, we obtain DE = \sqrt{51}.

Therefore, the length of DE is $\\boxed{\sqrt{51}}$. The answer is: \sqrt{51}.

The answer is: \sqrt{51}

Figure 14: Example 2 about Precalculus on the MATH dataset. GPT-4o results are available at GPT-4o results

Example 3 on GSM8K – Arithmetic

Question James decides to run 3 sprints 3 times a week. He runs 60 meters each sprint. How many total meters does he run a week?

Ground Truth

1. First, let's find out how many meters James runs each day.
*60 meters * 3 sprints = 180 meters*
2. Now that we know how many meters James runs each day, we can calculate how many meters he runs in a week.
*180 meters per day * 3 days = 540 meters per week*
3. So, the total number of meters that James runs every week is 540. The answer is: 540

$\boxed{540}$

Few-shot CoT response:

Let's think step by step.

1. First, let's find out how many meters James runs each day. 60 meters * 3 sprints = 180 meters

2. Now that we know how many meters James runs each day, we can calculate how many meters he runs in a week.

3. 180 meters per day * 3 days = 540 meters per week

4. So, the total number of meters that James runs every week is 540.

The answer is: $\boxed{540}$ ✗

HiAR-ICL response:

Step 1: One-Step Thought
Calculate how many meters James runs per day.
There are 3 sprints per day, and each sprint is 60 meters. So, James runs 3 x 60 = 180 meters per day.

Step 2: One-Step Thought
Obtain the total meters that James run every week.
Since this is done 3 times a week, he will run a total of 180 * 3 = 540 meters per week. So the answer is: $\boxed{540}$ ✓

Figure 15: Example 3 about arithmetic on the GSM8K dataset.

Example 4 on GSM8K – Arithmetic

Question After tracking his workout routine for 8 weeks, Josh discovered that he had spent 4 hours working out every week except on some occasions. At one point, he spent 5 hours each for two consecutive weeks. Another time, he spent 6 hours in one week. What is the total number of hours he has spent working out across the 8 weeks?

Ground Truth

1. For 2 weeks, he spent 5 hours each for a total of 5*2 = <<5*2=10>>10 hours
2. In another week, he spent 6 hours; which gives a total of 10+6 = <<10+6=16>>16 hours
3. 2+1 = <<2+1=3>>3 weeks have been accounted for
4. He tracked 8 weeks so there are 8-3 = <<8-3=5>>5 weeks left
5. He spent 4 hours each week for 5 weeks for a total of 4*5 = <<4*5=20>>20 hours
6. The total across the 8 weeks is: 16+20 = <<16+20=36>>36 hours

$\boxed{36}$

Few-shot CoT response:

Let's think step by step.
1. Josh worked out for 4 hours each week for 8 weeks, which is 4 * 8 = 32 hours.
2. Additionally, he worked out for 5 hours for two weeks, so that's 5 * 2 = 10 hours. And he also worked out for 6 hours in one week.
3. Therefore, the total additional hours are 10 + 6 = 16 hours. Adding these together, the total hours he spent working out is 32 + 16 = 48 hours.
4. The answer is: $\boxed{48}$ ✗

HiAR-ICL response:

Step 1: System Analysis
Condition 1: 8-week workout period tracked by Josh.
Condition 2: Except for specified exceptions, he worked out 4 hours weekly.
Condition 3: For two consecutive weeks, he increased his workout duration to 5 hours each week.
Condition 4: During another week, he worked out for 6 hours.
Question: Calculate the total number of hours Josh has spent working out over the course of these 8 weeks.?

Step 2: Chain-of-thought
Let's break it down step by step.
1. Out of the 8 weeks, there are 2 weeks where Josh exercised for 5 hours each, totaling 2 * 5 = 10 hours.
2. Additionally, during one week, he exercised for 6 hours.
3. Therefore, subtracting these weeks leaves us with 8 - 2 - 1 = 5 weeks where he followed the general rule of exercising 4 hours per week, totaling 5 * 4 = 20 hours.
4. Adding up all the exercise time gives us 10 + 6 + 20 = 36 hours. The answer is: $\boxed{36}$ ✓

Figure 16: Example 4 about arithmetic on the GSM8K dataset.

Example 5 on StrategyQA – Commonsense

Question Guitar Hero: video game series. Is Guitar Hero Beatles inappropriate for a US third grader?

Ground Truth

1. Guitar Hero: Beatles is generally considered appropriate for children, including U.S. third graders, in terms of its gameplay and content.
2. The game features songs by The Beatles, a legendary band with a broad cultural appeal, and the gameplay itself is based on rhythm and music, which is family-friendly.
3. However, there are some Beatles songs that might have lyrics that could be more suitable for older audiences, but none of them would be inappropriate for a third grader in terms of content or themes.

False

Few-shot CoT response:

Let's think step by step.

1. The Guitar Hero Beatles is rated T (Teen) by the ESRB for suggestive themes and brief mild lyrics.

2. US third graders are generally around 8-9 years old, which may be too young for content that might seem mature based on the rating system.

3. The maturity level recommended by the ESRB suggests that it might be better suited for older children and teenagers.

The answer is: True

HiAR-ICL response:

Step 1: System Analysis
Condition 1: Guitar Hero: Beatles is a video game based on the music of the iconic British band "The Beatles".
Condition 2: Third graders in the United States are generally aged between 8-9 years old.
Question: Calculate the total number of hours Josh has spent working out over the course of these 8 weeks.?

Step 2: Chain-of-thought
Let's break it down step by step.
1. Condition 1 indicates that \"Guitar Hero: Beatles\" includes songs from The Beatles' extensive catalog, which is family-friendly and suitable for young audiences
2. However, condition 2 specifies that a typical U.S. third grader is around 8-9 years old . Some later songs might contain mild references that could be confusing or inappropriate for very young children without explanation. Since third graders vary widely in maturity level, parental discretion is advised.
3. But overall, Guitar Hero: The Beatles is not considered inappropriate for this age group.

The answer is: False

Figure 17: Example 5 about commonsense on the StrategyQA dataset.