

Beyond Examples: Towards Automated Thought-level In-Context Reasoning for Large Language Models

Jinyang Wu^{1*†}, Mingkuan Feng^{1*}, Shuai Zhang^{1,‡}, Feihu Che⁵, Zhengqi Wen⁵,
Chonghua Liao¹, Ling Yang², Haoran Luo³, Zheng Lian⁴, Jianhua Tao^{1,5,‡}

¹Tsinghua University ²Princeton University

³Nanyang Technological University

⁴Institute of Automation, Chinese Academy of Sciences

⁵Beijing National Research Center for Information Science and Technology

wu-jy23@mails.tsinghua.edu.cn

Abstract

In-context learning (ICL) leverages demonstrations to enhance the performance of large language models (LLMs). However, traditional ICL struggles with complex reasoning mainly due to superficial, example-level implicit imitation. To address these limitations, we introduce **ThoughtICR**, an automated **Thought-level In-Context Reasoning** paradigm that shifts from surface-level examples to more guidance-oriented thought patterns. Specifically, we first define atomic reasoning actions and construct thought patterns on small-scale seed data using Monte Carlo Tree Search (MCTS). During inference, we dynamically select appropriate thought patterns based on target problem attributes, providing explicit guidance for model reasoning. Thanks to its automated and strategic design, our method enables seamless plug-and-play integration with various post-training techniques. Experimental results demonstrate that our method improves performance across different model sizes and generalizes effectively across reasoning domains. Using only small-scale seed data, we achieve 80.6% accuracy on MATH and 62.5% on AMC, surpassing GPT-4o’s 77.2% and 57.5%, respectively. Moreover, compared to test-time scaling methods, our approach reduces computational costs by over 10×. Our code is available at <https://github.com/jinyangwu/ThoughtICR>.

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities across diverse tasks and domains (Zhao et al., 2023; Mohammadi et al., 2025; Wu et al., 2026a). Their proficiency in complex reasoning serves as a critical benchmark for evaluating fundamental cognitive abilities (Xi et al., 2024; Besta et al., 2025). However, mastering such multi-

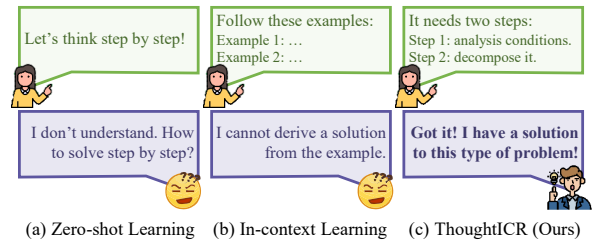


Figure 1: **Solution comparison.** (a) Zero-shot learning relies on a generic “think step by step” instruction, which is insufficient for detailed reasoning. (b) In-context learning needs carefully selected examples but struggles when faced with dissimilar problems. (c) ThoughtICR offers high-level reasoning patterns, ensuring robust performance across various problems.

step reasoning remains a significant challenge, requiring careful adherence to intricate rules and advanced problem-solving strategies.

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., 2024b). Rooted in analogy-based learning (Brown et al., 2020), ICL strategically leverages curated demonstrations to guide models in uncovering latent patterns and generating sophisticated reasoning trajectories. A key advancement in this area is Chain-of-Thought (CoT) (Wei et al., 2022), which employs prompts like “let’s think step by step” to facilitate structured problem-solving (Sprague et al., 2025). Building on this foundational technique, recent research mainly focuses on enhancing ICL performance through prompt instruction optimization (Ramnath et al., 2025), demonstration selection (Luo et al., 2024; Agarwal et al., 2025), and test-time scaling (TTS) (Agarwal et al., 2024; Xia et al., 2025).

Despite these advances, existing ICL paradigms face three key limitations: **(1) Surface-level Example-based Implicit Imitation.** Traditional ICL relies on superficial example-level mimicry rather than explicit reasoning guidance, forcing

* Equal Contribution

† Project Lead

‡ Corresponding Authors.

models to infer implicit patterns through imitation (Zhao et al., 2024). This surface-level mechanism fails to capture deeper reasoning structures and constrains cross-domain generalization. (2) **Sensitivity to Demonstration Characteristics.** LLMs’ reasoning performance is highly sensitive to demonstration characteristics such as quantity, ordering, and label distributions (Wang et al., 2023a; Cui et al., 2024), requiring reconstruction when encountering different task formats (Dong et al., 2024). (3) **Limited Automation and Scalability.** Effective ICL typically requires human-crafted demonstrations or carefully designed selection pipelines (Wang et al., 2024d). Moreover, while few-shot ICL achieves suboptimal performance, sophisticated test-time scaling (TTS) techniques incur substantial computational overhead, creating a performance-efficiency dilemma.

To address these challenges, we propose **ThoughtICR**, an automated **Thought-level In-Context Reasoning** paradigm (Figure 1). Our approach extends traditional ICL by redefining “context” from surface-level examples to higher-order thought patterns, shifting from example-based implicit imitation to pattern-based explicit reasoning guidance. ThoughtICR consists of two main components: **Offline Processing:** We introduce five atomic reasoning actions as fundamental building blocks: system analysis, one-step thought, chain-of-thought, divide and conquer, and self-reflection and refinement. These actions effectively cover the key behavioral patterns required for complex reasoning tasks. Using these actions and 200 seed samples, we apply MCTS to automatically derive high-level thought patterns. Each derived pattern is then encapsulated into a structured “thought card” for explicit inference guidance. **Online Inference:** We dynamically select appropriate thought cards based on problem attributes and guide model reasoning using the encapsulated thought patterns. This step incorporates verification through a process reward model and self-consistency checks to validate the final solution. Extensive experiments across **nine benchmarks and seven domains** demonstrate that ThoughtICR significantly outperforms traditional ICL methods from mathematical to agentic reasoning, even achieving comparable results to GPT-4o with a 7B backbone. Our main contributions are:

- **Novel ICL Insight:** We transcend traditional ICL by extending “context” from specific examples to higher-level abstract reasoning patterns,

advancing the frontier of ICL research.

- **Automated and Adaptive Reasoning:** We propose a reasoning framework that automatically constructs diverse reasoning patterns and adaptively applies them based on problem attributes.
- **Extensive Experiments and Discussion:** Using Qwen2.5-7B, our method achieves 80.6% accuracy on MATH and 62.5% on AMC, surpassing GPT-4o’s 77.2% and 57.5%, while reducing computational cost by over 10× compared to leading TTS methods. ThoughtICR also exhibits strong practical applications, including cross-domain generalization, plug-and-play integration with post-training techniques like GRPO, and efficient test-time scaling.

2 Related Work

In-Context Learning via Examples. In-context learning allows LLMs to implicitly acquire reasoning strategies from demonstrations without parameter updates (Zhou et al., 2024b; Dong et al., 2024). Approaches such as CoT (Wei et al., 2022) guide reasoning with prompts like “think step by step”, while self-consistency (Wang et al., 2023b) enhances accuracy by aggregating multiple reasoning paths. However, these methods predominantly rely on example-level analogical learning, which limits generalization and often necessitates expert intervention for complex reasoning tasks (Wang et al., 2023a; Yang et al., 2024c, 2025). In contrast, our approach emphasizes high-level reasoning patterns over specific examples, enabling automated and adaptive inference across diverse tasks without human intervention. Notably, this paradigm synergizes LLMs’ implicit reasoning with explicit external guidance, yielding robust performance.

Tree-based TTS. LLMs demonstrate strong capabilities but often struggle with complex reasoning tasks (Zhao et al., 2023). Tree-based TTS algorithms have proven effective in expanding search spaces and strengthening reasoning (Zhou et al., 2024a). Recent methods like Tree of Thought (Yao et al., 2023) and Graph of Thought (Besta et al., 2024) enable non-linear reasoning via multiple LLM queries, while advanced systems such as AlphaMath (Chen et al., 2024) and rStar (Qi et al., 2025) further enhance reasoning but at considerable computational cost. In contrast, ThoughtICR strategically frontloads computation and leverages

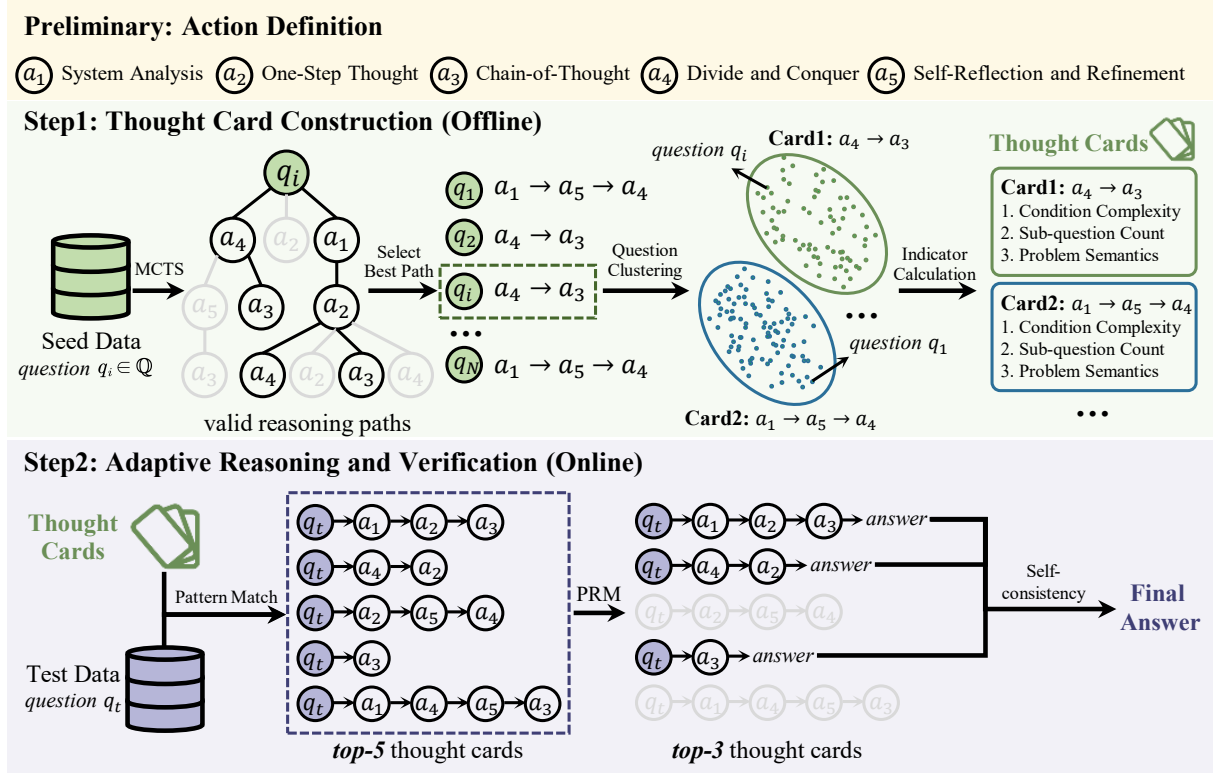


Figure 2: **Flowchart of our method ThoughtICR.** This framework consists of two main parts: (1) Thought Card Construction (Offline); and (2) Adaptive Reasoning and Verification (Online).

prior reasoning patterns, achieving competitive performance with substantially higher efficiency.

3 Methodology

Overview of ThoughtICR. As shown in Figure 2, our approach consists of two main components. The algorithm is detailed in Algorithm 1.

- *Step 1: Thought Card Construction (Offline):* Leverage MCTS to automatically construct high-level thought cards from seed data, which provide explicit guidance for subsequent inference.
- *Step 2: Adaptive Reasoning and Verification (Online):* Dynamically select optimal cards by problem attributes and verify candidate solutions.

3.1 Preliminary: Action Definition

Understanding human reasoning is crucial for modeling cognitive processes (Jaffe et al., 2023). Prior work (Kahneman, 2011; Da Silva, 2023) distinguishes two cognitive systems: *System 1*: fast and intuitive but error-prone; and *System 2*: slower yet more accurate. With the emergence of advanced models like OpenAI o1 and DeepSeek-R1, exploring efficient “System 2” reasoning has attracted growing attention (Qin et al., 2024; Guo et al.,

2025). 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). These actions capture essential behavioral patterns for complex reasoning (Wu et al., 2024a). Details are provided in Appendix C.1.

3.2 Step 1: Thought Card Construction

Based on the action definition, we introduce “*thought cards*”—structured reasoning templates that guide online inference in Step 2 (Sec. 3.3). From small-scale seed data \mathbb{Q} , we first obtain reasoning paths via MCTS (Phase 1), then abstract them into thought cards via clustering (Phase 2).

Phase 1: Acquiring Reasoning Paths via MCTS.

As shown in Figure 2, we employ MCTS to iteratively explore and generate high-quality reasoning trajectories for the seed dataset. This integrates MCTS’s exploration-exploitation balance with LLMs’ reasoning capabilities. The rationale for choosing MCTS over alternative algorithms is provided in Appendix A.6. Following Wang et al. (2025), we formulate each question $q_i \in \mathbb{Q}$ as a search problem, where q is the root node and sub-

sequent nodes represent reasoning steps (actions and corresponding outcomes) generated by policy LLM π_θ . We define state S_{t-1} as the trajectory q_i, s_1, \dots, s_{t-1} with $S_0 = q_i$. The next step is sampled as $s_t \sim \pi_\theta(S_{t-1})$. Each node s is assigned a reward $Q(s)$ (initialized to zero). When updated, the reward is propagated back to its parent p via:

$$Q(p) \leftarrow (1 - \alpha)Q(p) + \alpha Q(s) \quad (1)$$

where α is a discount factor. For terminal nodes, following Qi et al. (2025), we adopt majority voting likelihood as reward, enabling supervision-free generalization. MCTS involves four operations:

(1) *Selection*. This operation identifies promising nodes for expansion. Starting from the root node, we recursively select children until reaching a leaf node (maximum depth or answer node). To balance exploration and exploitation, we use the Upper Confidence Bounds for Trees (UCT):

$$UCT(s) = \begin{cases} Q(s) + w \sqrt{\frac{\ln N(p)}{N(s)}} & \text{if } N(s) > 0 \\ +\infty & \text{if } N(s) = 0 \end{cases} \quad (2)$$

where $N(s)$ is the visit count of node s , p is its parent, and w is exploration weight. The highest UCT node is selected for subsequent operations.

(2) *Expansion*. The chosen node s is expanded by sampling n actions from π_θ , and adding the resulting child nodes $\{s_1, s_2, \dots, s_n\}$ to the tree.

(3) *Simulation*. From s , we iteratively explore until reaching a leaf node s_d . To improve efficiency, we employ early termination when s_d 's self-consistency score $SC(s_d) > c$ for threshold $c \in [0, 1]$. This exploits that repeated actions at the same state likely indicate successful completion.

(4) *Backpropagation*. Node information is then updated along path s_0, \dots, s_d . Visit counts are incremented ($N(s) \leftarrow N(s) + 1$), and rewards $Q(s)$ propagate backward to parent p via Equation 1.

Phase 2: Abstracting Paths into Thought Cards.

After MCTS, each seed question yields multiple valid reasoning paths. We select the optimal path for each question, then cluster questions with the same thought patterns to construct thought cards.

Optimal Path Selection. We employ a Value of Computation (VoC) framework (Russell and Wefald, 1991) that balances benefits against costs. For each reasoning path p_{q_i} of question q_i , the score is:

$$\text{Score}(q_i, p_{q_i}) = k \cdot R(p_{q_i} | q_i) - (1 - k) \cdot C(p_{q_i}), \quad (3)$$

Algorithm 1 The Algorithm of ThoughtICR

Input: policy model π_θ ; test question q_t ; a small-scale seed set \mathbb{Q}

Output: the optimal reasoning trajectory y_t

- 1: Initialize action space $A = \{a_1, a_2, a_3, a_4, a_5\}$
 - 2: Initialize repository $D \leftarrow []$; Cards $\mathbb{C} \leftarrow \{\}$
 - 3: **for** $q_i \in \mathbb{Q}$ **do** \triangleright Step 1 (Offline)
 - 4: Acquire valid paths $P \leftarrow \text{MCTS}(\pi_\theta; q_i)$
 - 5: **if** $P \neq \emptyset$ **then**
 - 6: $p_{q_i, \text{best}} \leftarrow \arg \max_{p \in P} \text{Score}(q_i, p)$
 - 7: $D \leftarrow D \cup \{(q_i, p_{q_i, \text{best}})\}$
 - 8: **end if**
 - 9: **end for**
 - 10: $\mathbb{C} \leftarrow \text{Abstract}(D)$
 - 11: $\mathbb{C}_{q_t} \leftarrow \text{Card_Match}(q_t, \mathbb{C})$ \triangleright Step 2 (Online)
 - 12: $y_t \leftarrow \text{Reason_And_Verify}(\pi_\theta, q_t, \mathbb{C}_{q_t})$
-

where $R(p_{q_i} | q_i)$ is p_{q_i} 's final reward (defined as leaf node's Q-value), $C(p_{q_i})$ is reasoning cost (action count in p_{q_i}), and $k \in (0, 1)$ balances reward and cost. For each question q_i , the optimal path is:

$$p_{q_i, \text{best}} = \arg \max_{p_{q_i} \in P_{q_i}} \text{Score}(q_i, p_{q_i}). \quad (4)$$

This yields a *Question-Path Repository* $D = \{(q_i, p_{q_i, \text{best}}) | q_i \in \mathbb{Q}\}$ with one-to-one mappings between each seed question and its optimal path.

Thought Cards Abstraction via Clustering. Inspired by metareasoning (Russell and Wefald, 1991), we abstract $(q_i, p_{q_i, \text{best}})$ pairs from D into *thought cards* \mathbb{C} through clustering. Each card captures a high-level thought pattern (e.g., $a_1 \rightarrow a_2 \rightarrow a_3$), characterized by three commonly used indicators (Lee and Heyworth, 2000; Wu et al., 2024b): (1) *Condition Complexity (CC)*: prior condition count; (2) *Subquestion Count (SC)*: number of decomposed subproblems; (3) *Problem Semantics (PS)*: semantic embedding. Questions with identical thought patterns are clustered together to form a single thought card. Detailed descriptions are provided in Appendix C.4.

Card Representation. Finally, each thought card $c \in \mathbb{C}$ is defined as:

$$c = (\text{Pattern}(p_{\text{best}}), \overline{\text{CC}}, \overline{\text{SC}}, \overline{\text{PS}}), \quad (5)$$

where $\text{Pattern}(p_{\text{best}})$ is the abstract thought pattern, and $\overline{\text{CC}}, \overline{\text{SC}}, \overline{\text{PS}}$ are averaged indicators over all questions mapped to this pattern. We validate the effectiveness of these indicators in Section 4.6, with a card example in Figure 8 in the Appendix.

Table 1: **Performance comparison with ICL baselines.** We provide comprehensive results across nine benchmarks and seven domains. The best results in each box are highlighted in **bold**. All models are instruct versions.

Model	Method	Mathematics			Arith.	Science	Know.	Logic	Com.	Agent	Avg.
		MATH	AMC	AIME25	GSM8K	GPQA-D	MMLU-P	LogiQA	STG	BALROG	
GPT-4o	Zero-shot CoT	70.2	42.5	26.7	91.0	53.6	75.3	52.3	74.1	76.4	62.5
	Few-shot CoT	76.6	52.5	26.7	94.3	54.0	76.1	53.9	72.5	80.0	65.2
	CoT+SC@4	77.2	57.5	30.0	94.3	54.5	77.5	55.8	79.5	80.4	67.4
	Ours	84.8	62.5	36.7	96.0	58.1	83.9	62.4	82.2	85.6	72.5
Qwen2.5-14B	Zero-shot CoT	69.8	42.5	10.0	92.4	23.3	55.3	50.6	62.8	60.0	51.8
	Few-shot CoT	80.0	45.0	13.3	94.8	28.3	57.9	52.2	53.1	70.0	54.9
	CoT+SC@4	76.2	60.0	16.7	94.0	32.3	60.1	54.4	69.7	68.0	59.0
	Ours	81.4	65.0	23.3	95.8	44.4	66.7	61.9	77.3	76.8	66.2
Qwen2.5-7B	Zero-shot CoT	64.8	32.5	6.7	86.2	24.3	44.7	40.4	52.8	40.8	43.7
	Few-shot CoT	68.6	47.5	6.7	91.6	34.3	46.7	42.5	67.6	46.4	50.2
	CoT+SC@4	76.4	50.0	10.0	92.0	34.3	49.2	45.0	73.2	47.6	53.0
	Ours	80.6	62.5	20.0	93.7	43.4	57.5	54.4	76.0	56.0	60.5
Qwen2-7B	Zero-shot CoT	36.9	12.5	0.0	76.6	19.7	37.5	38.2	55.3	28.0	33.8
	Few-shot CoT	52.9	17.5	3.3	85.7	19.7	41.3	40.5	62.3	30.6	39.3
	CoT+SC@4	55.6	20.0	3.3	87.7	26.7	42.9	43.3	65.5	35.4	42.3
	Ours	66.8	30.0	10.0	91.8	40.4	48.3	52.1	72.0	43.2	50.5
Llama3-8B	Zero-shot CoT	5.8	5.0	0.0	68.3	11.1	30.9	34.9	57.2	33.8	27.4
	Few-shot CoT	17.8	7.5	0.0	74.5	22.7	35.9	39.1	68.4	36.8	33.6
	CoT+SC@4	28.8	15.0	0.0	80.6	24.7	41.8	44.0	66.8	37.6	37.7
	Ours	46.6	30.0	6.7	89.6	39.4	50.3	50.8	73.4	45.0	48.0
DeepSeek-Dis.-1.5B	Zero-shot CoT	78.4	20.0	16.7	77.3	23.6	53.2	47.3	60.9	30.0	45.3
	Few-shot CoT	80.4	25.0	20.0	80.5	27.8	54.6	50.5	61.3	36.8	48.6
	CoT+SC@4	82.6	37.5	20.0	84.9	32.3	58.3	53.7	67.5	37.0	52.7
	Ours	87.8	52.5	30.0	91.3	40.1	67.2	61.2	73.4	42.8	60.7

3.3 Step 2: Adaptive Reasoning and Verification

Adaptive Reasoning. During inference, given a test question q_t , we compute its CC, SC, and PS metrics using a model, and retrieve the top-5 most relevant thought cards via pattern matching based on nearest neighbor search (Muja and Lowe, 2014):

$$NN_5(q_t, \mathbb{C}) = \arg \min_{\mathbb{C}_{q_t} \subseteq \mathbb{C}, |\mathbb{C}_{q_t}|=5} \sum_{c \in \mathbb{C}_{q_t}} d(q_t, c) \quad (6)$$

where $NN_5(q_t, \mathbb{C})$ denotes five nearest cards under distance function d . For CC/SC, we use absolute differences (e.g., $|CC_{q_t} - CC_c|$); and for PS, we use cosine similarity between q_t 's semantic embeddings and each card's associated questions. The selected cards serve as templates that guide the model to execute corresponding actions sequentially.

Verification. To identify the most accurate solution, we introduce a two-stage process: an off-the-shelf PRM scores all reasoning paths under the guidance of top-5 cards, then the top-3 highest-scoring candidates undergo self-consistency (SC) checks for final selection. Interestingly, our experiments show that even SC-only verification effectively identifies precise reasoning paths. Detailed descriptions are provided in Appendix B.5.

4 Experiments

This section presents experimental settings, results, and analysis. We analyze from five aspects: performance, efficiency, cross-domain generalization, practical applications, and ablation study.

4.1 Experimental Settings

Datasets. Our evaluation spans seven reasoning domains and nine datasets: (1) **mathematics**: MATH (Hendrycks et al., 2021), AMC, AIME25 (Li et al., 2025); (2) **arithmetic (arith.)**: GSM8K (Cobbe et al., 2021); (3) **graduate-level science**: GPQA-Diamond (GPQA-D) (Rein et al., 2024); (4) **general knowledge (know.)**: MMLU-ProX (MMLU-P) (Xuan et al., 2025); (5) **logic**: LogiQA-v2 (Liu et al., 2023); (6) **commonsense (com.)**: StrategyQA (STG) (Geva et al., 2021); (7) **agent**: BALROG (Paglieri et al., 2025). While Table 1 reports results on all datasets, our analysis focuses on MATH, GSM8K, STG, AMC to ensure comparability with baselines that lack results on others (e.g., AIME). Details are in Appendix D.2.

Baselines. We evaluate ThoughtICR against three representative categories: (1) traditional example-based ICL, including zero-shot CoT (Kojima et al., 2022), few-shot CoT (Wei et al., 2022),

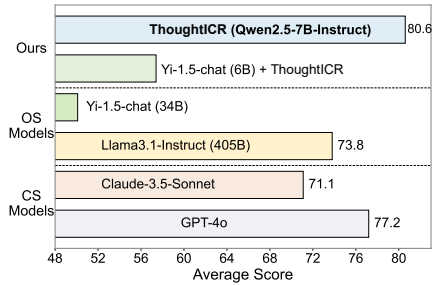


Figure 3: **Comparison with powerful LLMs on MATH.** ‘OS’ and ‘CS’ denote open-source and closed-source models.

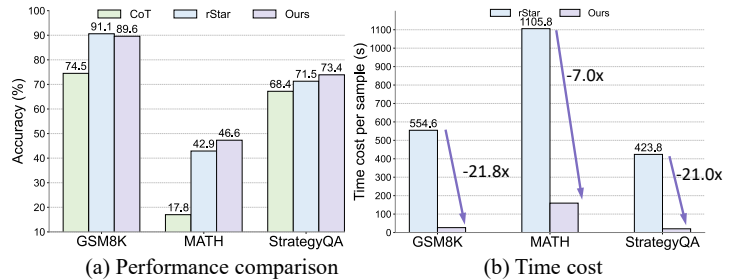


Figure 4: **Comparison with leading tree-based TTS method, rStar.** (a) Performance Comparison. (b) Total Time Cost Per Sample (the “end-to-end” time cost per sample).

Table 2: **Performance comparison with tree-based TTS methods.** The best results are highlighted in **bold**, with baseline results sourced from the original papers.

Model	Method	MATH	GSM8K	STG	Avg.
Qwen2-7B	ToT	53.3	79.0	66.7	66.3
	RAP	51.6	82.1	67.3	67.0
	ReST-MCTS*	52.4	82.3	64.9	66.5
	Ours	66.8	91.8	80.9	79.8
Llama3-8B	ToT	13.6	69.0	60.4	47.7
	RAP	18.8	80.5	68.7	56.0
	ReST-MCTS*	34.2	75.5	65.0	58.2
	LLaMA-Berry	39.6	88.1	-	-
	rStar	42.9	91.1	71.5	68.5
	Ours	46.6	89.6	73.4	69.9

and CoT+SC (Wang et al., 2023b); (2) tree-based TTS methods, including ToT (Yao et al., 2023), RAP (Hao et al., 2023), ReST-MCTS* (Zhang et al., 2024a), LLaMA-Berry (Zhang et al., 2025), and rStar (Qi et al., 2025); (3) powerful LLMs like GPT-4o (OpenAI, 2024). We provide more advanced baselines like 10-shot Reinforced ICL (Agarwal et al., 2024) in Appendix E.1.

Evaluation Metrics. In line with prior studies (Wang et al., 2025; Qi et al., 2025), we report two main metrics: *accuracy* and *average time cost per sample*. More details are in Appendix D.5.

Implementation Details. For GSM8K, MATH, and STG, we use 200 randomly sampled training instances as seed data and evaluate on respective test sets. For other benchmarks, we reuse 200 MATH samples as seed data (details in Appendix D.6). Note that, unless otherwise specified, Llama3-8B-Instruct serves as the default policy LLM π_θ for MCTS-based thought card construction in our main experiments. We verified that using alternative policy LLMs (e.g., Qwen2.5-7B-Instruct) yields similar thought card quality and final reasoning performance, indicating that ThoughtICR is robust to the choice of policy LLM rather than being model-

dependent.

4.2 Performance Comparison with Baselines

Main Comparison with ICL. Table 1 presents the main results across different models, datasets, and domains. We have three key findings:

◇ *ThoughtICR Consistently Outperforms Traditional ICL Baselines.* ThoughtICR achieves substantial improvements across all benchmarks and models. For instance, Llama3-8B improves from 17.8% (Few-shot CoT) to 46.6% on MATH (2.6 \times gain) and from 36.8% to 45.0% on BALROG. This demonstrates the effectiveness of thought-level guidance over example-level imitation.

◇ *ThoughtICR Generalizes Across Diverse Domains.* Despite constructing thought cards mainly from mathematical seed data, our method generalizes effectively to diverse domains, including science (GPQA-D), logic (LogiQA), and agentic tasks (BALROG). This validates that high-level thought patterns capture general, domain-agnostic reasoning structures rather than task-specific features.

◇ *ThoughtICR Integrates Seamlessly with TTS.* Our approach incorporates self-consistency and Best-of-N strategy during inference, demonstrating integration potential with mainstream TTS techniques. This compatibility suggests promising future directions for hybrid methods that combine thought-level guidance with test-time scaling.

Comparison with Tree-based TTS. Table 2 compares ThoughtICR with representative tree-based TTS methods. Our approach consistently outperforms baselines across models and datasets, with notable gains on complex datasets like MATH. For example, ThoughtICR achieves 79.8% average accuracy versus the best baseline’s 67.0% on Qwen2-7B. Unlike traditional TTS methods that rely on broad exploration, ThoughtICR leverages offline thought cards to guide inference adap-

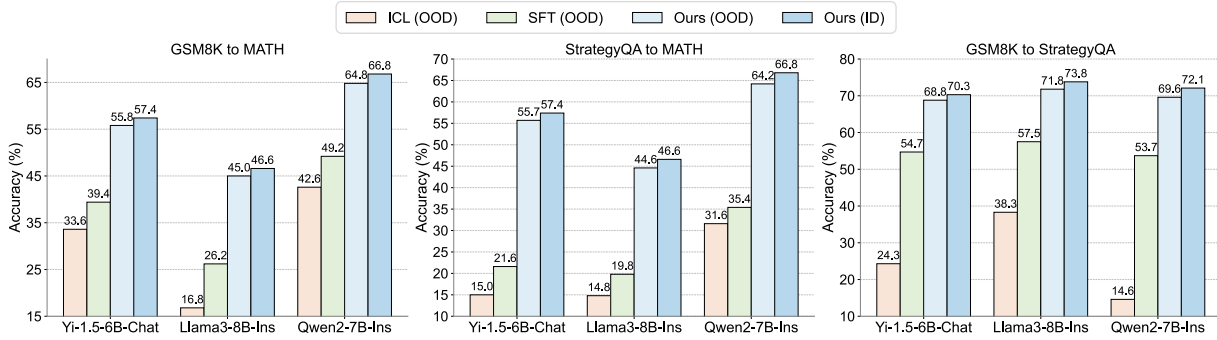


Figure 5: **Cross-domain generalization results.** “ID” and “OOD” denote in-distribution and out-of-distribution, respectively. “A to B” represents constructing cards from dataset A and testing on B. ThoughtICR shows superior OOD generalization compared with example-based ICL and SFT methods.

Table 3: **Plug-and-play verification.** ThoughtICR further enhances GRPO-trained Qwen2.5-7B model.

Method	MATH	AMC	GPQA-D	Avg.
Base	64.8	32.5	24.3	40.5
+GRPO	78.2	55.0	38.8	57.3
+ThoughtICR	83.4	65.0	44.0	64.2
Δ (\uparrow)	6.7%	18.2%	13.4%	12.1%

Table 4: **Scalability verification.** We provide performance with varying seed data sizes on Qwen2.5-7B.

Seed Data Size	100	200	500	1000
Math Accuracy (%)	76.6	80.6	82.2	83.0
AMC Accuracy (%)	55.0	62.5	65.0	67.5
GPQA-D Accuracy (%)	37.9	43.4	44.9	45.9
Avg. Accuracy (%)	56.5	62.2	64.1	65.5

tively, achieving higher efficiency and accuracy. We provide more performance comparison in Appendix E.1 and efficiency analysis in Figure 4.

Comparison with Powerful LLMs. As shown in Figure 3, Qwen2.5-7B with ThoughtICR achieves 80.6% accuracy on MATH, outperforming GPT-4o (77.2%). Similarly, Yi-1.5-6B with ThoughtICR surpasses Yi-1.5-34B. This demonstrates that ThoughtICR-empowered smaller models can exceed much larger models. Unlike traditional ICL’s implicit example-based imitation, ThoughtICR provides explicit high-level thought patterns that equip smaller models with advanced problem decomposition abilities typically found in larger models, thus enabling superior performance. Additional results are provided in Table 12 of the Appendix.

4.3 Computational Efficiency Analysis

We compare ThoughtICR with rStar (Qi et al., 2025), a leading tree-based TTS method that extensively explores the solution space. Figure 4 presents both performance and computational efficiency comparisons. Our analysis uses comprehensive “end-to-end” time costs per sample, encompassing both offline thought card construction and online inference (details in Appendix E.7).

ThoughtICR achieves competitive performance with rStar while substantially reducing computational overhead: 21.8 \times speedup on GSM8K,

21.0 \times on StrategyQA, and 7.0 \times even on the challenging MATH benchmark. These gains stem from efficient online inference that adaptively selects appropriate thought patterns rather than exhaustively searching the solution space. This demonstrates ThoughtICR’s ability to balance performance and efficiency, making advanced TTS reasoning more practical for real-world deployment.

4.4 Cross-Domain Generalization Analysis

Recent studies highlight the critical impact of distributional bias on LLMs’ reliability (Yuan et al., 2023; Wang et al., 2024a). Despite strong in-distribution (ID) performance, models often underperform on out-of-distribution (OOD) data (Yang et al., 2024b), compounded by the difficulty of acquiring sufficient OOD samples.

We evaluate ThoughtICR against ICL and supervised fine-tuning (SFT) under OOD scenarios. For fair comparison, we use the same 200 seed samples for both thought card construction and SFT. As shown in Figure 5, while ICL and SFT experience significant performance degradation, ThoughtICR demonstrates remarkable resilience across multiple models and datasets. This robustness stems from ThoughtICR’s automated construction process and high-level pattern-based guidance, which together capture general reasoning structures without human intervention or task-specific features. We further explore weak-to-strong generalization in Ap-

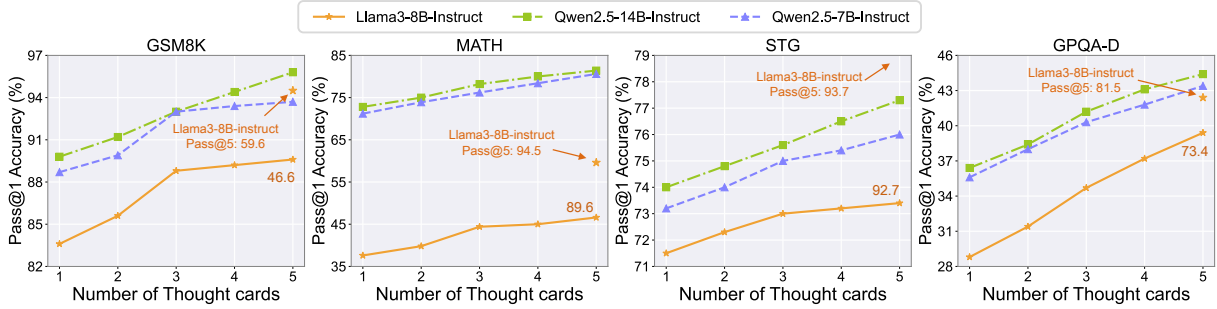


Figure 6: **Test-time scaling law verification.** We examine the variation in ThoughtICR performance with the number of selected reasoning guidance-providing thought cards in Section 3.3.

Table 5: **Ablation study.** We report the performance on Llama3-8B after removing or replacing each component of ThoughtICR. All modules prove essential for optimal performance.

Model Setting	Module	GSM8K	MATH	STG	AMC	GPQA-D	MMLU-P	Average	Δ (\downarrow)
ThoughtICR (Full)	-	89.6	46.6	73.4	30.0	39.4	50.3	54.9	-
- w/o T.C.	Sec. 3.2	79.9	33.4	63.3	12.5	22.7	35.9	41.3	-13.6
- w/o C.M.	Sec. 3.3	84.1	39.2	68.0	20.0	29.3	41.2	46.4	-8.5
- w/ C.M. (subquestion)	Sec. 3.3	86.9	42.0	71.5	27.5	35.6	47.9	51.9	-3.0
- w/ C.M. (semantic)	Sec. 3.3	86.4	40.2	70.6	25.0	33.2	43.2	49.8	-5.1
- w/o F.V.	Sec. 3.3	83.3	40.2	69.6	25.0	32.9	45.2	49.4	-5.5
- w/ F.V. (SC)	Sec. 3.3	87.3	42.6	70.3	27.5	36.9	47.8	52.1	-2.8
- w/ F.V. (PRM)	Sec. 3.3	88.8	43.8	73.4	30.0	37.3	48.5	53.6	-1.3

pendix E.3, demonstrating that our method can even enhance GPT-4o’s performance using guidance from weaker models (e.g., Llama2-7B).

4.5 Practical Applicability Analysis

As shown in Table 3, 4, and Figure 6, we examine ThoughtICR’s compatibility with existing training paradigms and flexible scalability across real-world deployment scenarios.

Integration with Post-Training. Similar to ICL methods like CoT, ThoughtICR operates as a flexible framework compatible with post-training techniques. We applied ThoughtICR to Qwen2.5-7B with GRPO training on MATH. As shown in Table 3, our framework further boosts performance, improving average accuracy from 57.3% (GRPO alone) to 64.2% (GRPO+ThoughtICR). This synergy suggests that ThoughtICR captures reasoning patterns complementary to those learned during post-training, confirming its plug-and-play versatility. Additional results are in Appendix E.5.

Thought Card Library Scalability. Table 4 analyzes performance sensitivity to seed dataset size (100 to 1K samples). Results show a clear positive correlation: average performance consistently improves from 56.5% to 65.5% as seed data increases. This confirms that our thought card library is expandable: larger seed datasets enrich the library

and provide more comprehensive reasoning guidance. We use 200 samples by default to balance performance and computational cost.

Test-Time Scaling. We investigate whether our method follows test-time scaling laws by varying the number of selected thought cards during inference. As shown in Figure 6, progressively adding more thought cards consistently improves performance across all models and benchmarks. This scaling behavior opens opportunities for adaptive mechanisms that dynamically determine the optimal number of cards based on task requirements.

4.6 Ablation Study and Other Analysis

Ablation Study. As shown in Table 5, we analyze three core components of ThoughtICR by replacing: thought cards (T.C.) with random action combinations; card matching (C.M.) with random selection, subquestion, and semantic metric; and final verification (F.V.) with random selection, self-consistency, and PRM. We observe that removing any component results in performance degradation.

Other Analysis. We include further results and analysis in Appendix: performance comparisons (E.1), combination with reasoning models (E.2), weak-to-strong generalization (E.3), multi-task systems (E.4), SFT integration (E.5), accuracy across problem complexity (E.6), extended ablation (E.9),

statistical analysis (E.12), and case study (G).

5 Conclusion

We propose ThoughtICR, an automated reasoning paradigm that advances ICL from example-based imitation to thought-level explicit guidance. By employing adaptive, high-level strategies, ThoughtICR enables LLMs to develop genuine reasoning abilities and show impressive performance across domains, datasets, and models. This work establishes foundations for generalizable reasoning frameworks and opens directions for constructing problem-solving patterns in other domains.

Limitations

Despite ThoughtICR’s promising performance, several limitations remain, providing opportunities for future research directions.

First, our experimental validation primarily focuses on textual reasoning tasks, leaving open questions about the paradigm’s applicability to multimodal reasoning scenarios. Extending ThoughtICR to visual-linguistic reasoning contexts and more realistic agentic tool (skill) use scenarios (Mohammadi et al., 2025; Wu et al., 2026b; Xia et al., 2026) represents a compelling avenue for future exploration.

Second, as a flexible training-free framework, ThoughtICR significantly enhances the reasoning capabilities of relatively weak models. Future work could investigate integrating ThoughtICR with more post-training methods, enabling more sophisticated reasoning patterns that dynamically adapt throughout model optimization.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. U21B2010, No. 62322120), and the China Postdoctoral Science Foundation (Grant No. 2025T180461 and 2025M771685).

References

Ishika Agarwal, Krishnateja Killamsetty, Lucian Popa, and Marina Danilevsky. 2025. DELIFT: Data efficient language model instruction fine-tuning. In *The Thirteenth International Conference on Learning Representations*.

Rishabh Agarwal, Avi Singh, Lei M Zhang, Bernd Bohnet, and 1 others. 2024. Many-shot in-context

learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

- Anthropic. 2024. [Introducing claude 3.5 sonnet](#).
- Susan M Barnett and Stephen J Ceci. 2002. When and where do we apply what we learn?: A taxonomy for far transfer. *Psychological bulletin*, 128(4):612.
- Maciej Besta, Julia Barth, Eric Schreiber, Ales Kubicek, Afonso Catarino, Robert Gerstenberger, Piotr Nyczyk, Patrick Iff, Yueling Li, Sam Houlliston, and 1 others. 2025. Reasoning language models: A blueprint. *arXiv preprint arXiv:2501.11223*.
- Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and 1 others. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690.
- 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, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Eric Sigler, Mateusz Litwin, Scott Gray, and 7 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.
- 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*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- 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.
- 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, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Luyu Gao, Aman Madaan, Shuyan Zhou, and 1 others. 2023. Pal: program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*. JMLR.org.
- 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](#).
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shitong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- 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)*.
- Keith J Holyoak and Kyunghee Koh. 1987. Surface and structural similarity in analogical transfer. *Memory & cognition*, 15(4):332–340.
- 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.
- 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.
- Samuel Kotz and Norman L. Johnson, editors. 1992. *Breakthroughs in Statistics: Methodology and Distribution*. Springer New York, New York, NY.
- Fong-Lok Lee and Rex Heyworth. 2000. Problem complexity: A measure of problem difficulty in algebra by using computer. *Education Journal*, 28(1):85–108.
- Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q. Jiang, Ziju Shen, and 1 others. 2025. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. <https://huggingface.co/datasets/Numinamath>. Hugging Face repository, 13:9.
- 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*.
- Hanmeng Liu, Jian Liu, Leyang Cui, Zhiyang Teng, Nan Duan, Ming Zhou, and Yue Zhang. 2023. Logiqa 2.0 - an improved dataset for logical reasoning in natural language understanding. *IEEE ACM Trans. Audio Speech Lang. Process.*, 31:2947–2962.
- Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.
- Jianqiao Lu, Zhiyang Dou, Hongru WANG, and 1 others. 2024. AutoPSV: Automated process-supervised verifier. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- 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](#).
- Mahmoud Mohammadi, Yipeng Li, Jane Lo, and Wendy Yip. 2025. Evaluation and benchmarking of llm agents: A survey. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 6129–6139.
- Marius Muja and David G. Lowe. 2014. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2227–2240.
- OpenAI. 2024. [Hello gpt-4o](#).
- Davide Paglieri, Bartłomiej Cupiał, Samuel Coward, Ulyana Piterbarg, Maciej Wolczyk, Akbir Khan, Eduardo Pignatelli, Łukasz Kuciński, Lerrel Pinto, Rob Fergus, Jakob Nicolaus Foerster, Jack Parker-Holder, and Tim Rocktäschel. 2025. BALROG: Benchmarking agentic LLM and VLM reasoning on games. In *The Thirteenth International Conference on Learning Representations*.

- 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*.
- Zhenting Qi, Mingyuan MA, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. 2025. Mutual reasoning makes smaller LLMs stronger problem-solver. In *The Thirteenth International Conference on Learning Representations*.
- Yiwei Qin, Xuefeng Li, Haoyang Zou, Yixiu Liu, Shijie Xia, Zhen Huang, Yixin Ye, Weizhe Yuan, Hector Liu, Yuanzhi Li, and 1 others. 2024. O1 replication journey: A strategic progress report–part 1. *arXiv preprint arXiv:2410.18982*.
- Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. 2024. Recursive introspection: Teaching language model agents how to self-improve. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Qwen Team. 2024. [Qwen2.5: A party of foundation models](#).
- Kiran Ramnath, Kang Zhou, Sheng Guan, Soumya Smruti Mishra, Xuan Qi, Zhengyuan Shen, Shuai Wang, Sangmin Woo, Sullam Jeoung, Yawei Wang, and 1 others. 2025. A systematic survey of automatic prompt optimization techniques. *arXiv preprint arXiv:2502.16923*.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.
- Stuart Russell and Eric Wefald. 1991. Principles of metareasoning. *Artificial Intelligence*, 49(1):361–395.
- Fobo Shi, Peijun Qing, Dong Yang, Nan Wang, Youbo Lei, Haonan Lu, Xiaodong Lin, and Duantengchuan Li. 2024. Prompt space optimizing few-shot reasoning success with large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1836–1862, Mexico City, Mexico. Association for Computational Linguistics.
- Zayne Rea Sprague, Fangcong Yin, Juan Diego Rodriguez, and 1 others. 2025. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning. In *The Thirteenth International Conference on Learning Representations*.
- Linzhuang Sun, Hao Liang, Jingxuan Wei, Bihui Yu, Conghui He, Zenan Zhou, and Wentao Zhang. 2024. Beats: Optimizing llm mathematical capabilities with backverify and adaptive disambiguate based efficient tree search. *arXiv preprint arXiv:2409.17972*.
- Mirac Suzgun and Adam Tauman Kalai. 2024. Meta-prompting: Enhancing language models with task-agnostic scaffolding. *arXiv preprint arXiv:2401.12954*.
- 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*.
- Ante Wang, Linfeng Song, Ye Tian, Baolin Peng, Dian Yu, Haitao Mi, Jinsong Su, and Dong Yu. 2025. Litesearch: Efficient tree search with dynamic exploration budget for math reasoning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(24):25318–25326.
- 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. 2024a. 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. 2024b. 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.
- Peiyi Wang, Lei Li, Zhihong Shao, and 1 others. 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.
- 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*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.
- Jinyang Wu, Chonghua Liao, Mingkuan Feng, Shuai Zhang, Zhengqi Wen, Haoran Luo, Ling Yang,

- Huazhe Xu, and Jianhua Tao. 2025. Templaterl: Structured template-guided reinforcement learning for llm reasoning. *arXiv preprint arXiv:2505.15692*.
- Jinyang Wu, Shuo Yang, Changpeng Yang, Yuhao Shen, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. 2026a. Spark: Strategic policy-aware exploration via dynamic branching for long-horizon agentic learning. *arXiv preprint arXiv:2601.20209*.
- Jinyang Wu, Guocheng Zhai, Ruihan Jin, Jiahao Yuan, Yuhao Shen, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. 2026b. Atlas: Orchestrating heterogeneous models and tools for multi-domain complex reasoning. *arXiv preprint arXiv:2601.03872*.
- Siwei Wu, Zhongyuan Peng, Xinrun Du, Tuney Zheng, Minghao Liu, Jialong Wu, Jiachen Ma, Yizhi Li, Jian Yang, Wangchunshu Zhou, and 1 others. 2024a. A comparative study on reasoning patterns of openai’s o1 model. *arXiv preprint arXiv:2410.13639*.
- Zhuofeng Wu, Richard He Bai, Aonan Zhang, Jiatao Gu, V.G.Vinod Vydiswaran, Navdeep Jaitly, and Yizhe Zhang. 2024b. Divide-or-conquer? which part should you distill your LLM? In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 2572–2585, Miami, Florida, USA. Association for Computational Linguistics.
- 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, and 2 others. 2024. Training large language models for reasoning through reverse curriculum reinforcement learning. In *Forty-first International Conference on Machine Learning*.
- Fanzeng Xia, Yidong Luo, Tinko Sebastian Bartels, Yaqi Xu, and Tongxin Li. 2025. Rethinking the unsolvable: When in-context search meets test-time scaling. *arXiv preprint arXiv:2505.22290*.
- Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, and 1 others. 2026. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*.
- Yuxi Xie, Anirudh Goyal, Wenyue Zheng, and 1 others. 2024. Monte carlo tree search boosts reasoning via iterative preference learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Weihao Xuan, Rui Yang, Heli Qi, Qingcheng Zeng, Yunze Xiao, Aosong Feng, Dairui Liu, Yun Xing, Junjue Wang, Fan Gao, and 1 others. 2025. Mmlu-prox: A multilingual benchmark for advanced large language model evaluation. *arXiv preprint arXiv:2503.10497*.
- Hang Yan, Fangzhi Xu, Rongman Xu, Yifei Li, Jian Zhang, Haoran Luo, Xiaobao Wu, Luu Anh Tuan, Haiteng Zhao, Qika Lin, and 1 others. 2025. Mur: Momentum uncertainty guided reasoning for large language models. *arXiv preprint arXiv:2507.14958*.
- An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, and 1 others. 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*.
- Ling Yang, Zhaochen Yu, Tianjun Zhang, Minkai Xu, Joseph E. Gonzalez, Bin CUI, and Shuicheng YAN. 2025. Supercorrect: Advancing small LLM reasoning with thought template distillation and self-correction. In *The Thirteenth International Conference on Learning Representations*.
- Yuqing Yang, Yan Ma, and Pengfei Liu. 2024d. Weak-to-strong reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 8350–8367, Miami, Florida, USA. Association for Computational Linguistics.
- 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.
- Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, and 1 others. 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*.
- Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. 2025. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*.

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, and 1 others. 2025. LLaMA-berry: Pairwise optimization for olympiad-level mathematical reasoning via o1-like Monte Carlo tree search. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7315–7337, Albuquerque, New Mexico. Association for Computational Linguistics.

Xuan Zhang, Chao Du, Tianyu Pang, Qian Liu, Wei Gao, and Min Lin. 2024c. Chain of preference optimization: Improving chain-of-thought reasoning in LLMs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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, and 1 others. 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*.

Yuxiang Zhou, Jiazheng Li, Yanzheng Xiang, Hanqi Yan, Lin Gui, and Yulan He. 2024b. 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 ThoughtICR

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

Contents

A	Further Discussions on ThoughtICR	14
A.1	Discussion on ThoughtICR and ICL	14
A.2	Discussion on ThoughtICR and RLVR	15
A.3	Theoretical Analysis	15
A.4	Failure mode analysis of pattern-based approach	15
A.5	Human Interpretability	16
A.6	Rationale for Employing Monte Carlo Tree Search	16
B	Preliminaries	17
B.1	Overall Notations	17
B.2	LLM Reasoning	17
B.3	In-Context Learning	18
B.4	Monte Carlo Tree Search	19
B.5	Verification Methods	20
C	Algorithm Details	20
C.1	Action Space	20
C.2	Reward Value in MCTS	21
C.3	Thought Card Visualization	21
C.4	Card Matching	21
C.5	Verification	22
D	Detailed Experimental Setups	22
D.1	Models	22
D.2	Datasets	23
D.3	Card Distribution	25
D.4	Baselines	26
D.5	Evaluation Details	26
D.6	Implementation Details	26
E	Supplementary Results	26
E.1	Detailed Comparison with Powerful LLMs and Baseline Methods	26
E.2	Combination with Large Reasoning Models	27
E.3	Weak-to-Strong Generalization	28
E.4	Multi-Task System	28
E.5	Integration with SFT	30
E.6	Performance across Problem Complexity	30
E.7	Detailed Computational Cost	30
E.8	Effective and Efficient TTS	30
E.9	Additional Ablation Study	31
E.10	Thought Pattern Analysis	32
E.11	Hyperparameter Sensitivity	33
E.12	Statistical Validation	33
F	Prompts	34

In this section, we provide a detailed discussion on ThoughtICR and ICL A.1, theoretical analysis A.3, failure mode analysis of pattern-based approach A.4, human interpretability A.5, and the rationale for employing MCTS to construct thought cards A.6.

A.1 Discussion on ThoughtICR and ICL

In this work, we introduce ThoughtICR, a novel paradigm within the broader in-context learning (ICL) framework. Originally proposed by prior work (Brown et al., 2020), ICL refers to *a process where the model is provided with natural language instructions and/or task examples and is expected to complete new task instances by predicting subsequent content*. Our proposed ThoughtICR paradigm maintains alignment with this foundational definition while extending its scope through two main dimensions:

Paradigm Adherence. Although ThoughtICR introduces a distinct pre-computation phase through MCTS to systematically construct high-level thought patterns (termed “thought cards”), the paradigm fundamentally remains within the ICL framework. During inference, ThoughtICR leverages these generated thought patterns explicitly formulated as natural language instructions. This approach retains the core mechanism of guiding the model via linguistic context, thus adhering strictly to the traditional conceptualization of ICL despite the augmented preparatory step.

Context Redefinition. We significantly expand the conventional interpretation of “context” within ICL. Traditionally, “context” refers strictly to task-specific demonstration examples. In contrast, ThoughtICR shifts the focus from concrete example-driven contexts to more abstract, generalized thought patterns. This redefinition allows context to transcend task-specific boundaries, enabling models to generalize reasoning strategies effectively across diverse domains and tasks. Consequently, our broader definition enhances cross-task adaptability, significantly enriching the concept of context originally outlined by prior work (Brown et al., 2020).

By redefining context from example-specific instances to higher-level reasoning abstractions, ThoughtICR provides a robust, generalizable foundation for enhancing LLM performance, especially

on complex reasoning tasks. This paradigm extension paves the way for future research that explores richer, more flexible contexts, facilitating stronger generalization capabilities across various problem-solving scenarios.

A.2 Discussion on ThoughtICR and RLVR

RLVR-based methods (Guo et al., 2025; Yue et al., 2025; Wu et al., 2025, 2026a) improve reasoning by internalizing correct reasoning chains through reinforcement learning training, thereby enhancing the model’s intrinsic reasoning probability. In contrast, ThoughtICR is training-free and improves reasoning by providing explicit external strategic guidance in the form of thought patterns constructed via MCTS. These two paradigms differ in two key dimensions: (1) **Methodology**: ThoughtICR requires no parameter updates and operates purely at inference time, while RLVR requires costly reinforcement learning training. (2) **Source of improvement**: ThoughtICR combines external guidance priors with the model’s intrinsic reasoning, whereas RLVR directly amplifies the probability of correct internal reasoning chains. Crucially, these approaches are *complementary*: ThoughtICR can be applied on top of RLVR-trained models (e.g., GRPO, DeepSeek-R1, OpenAI-o1) to provide additional strategic scaffolding, consistently yielding further performance gains (see Table 15 and Table 16).

A.3 Theoretical Analysis

To further understand the advantages of our approach, we provide a theoretical analysis of *why* and *when* pattern-based methods outperform traditional example-based methods.

Why Pattern-Based Methods Are More Effective Than Example-Based Methods. The superiority of our method can be attributed to three core principles:

Pattern Extraction vs. Implicit Learning.

Our pattern-based approach explicitly extracts verifiable thought patterns from examples using Monte Carlo Tree Search (MCTS), which helps ensure the correctness of the learned logic. In contrast, standard example-based methods rely on implicit pattern learning. This form of implicit analogical reasoning can often produce incorrect or flawed patterns, particularly in complex multi-step reasoning tasks.

Dual-Process Theory and Explicit Reasoning.

Following Kahneman’s dual-process theory (Kahneman, 2011), complex reasoning is a function of deliberative System 2 processing, not intuitive System 1 thinking. Example-based In-Context Learning (ICL) often leans on implicit analogical reasoning, which is prone to the surface-level matching errors characteristic of System 1 (Holyoak and Koh, 1987). Our method, however, provides structured reasoning scaffolding through atomic actions. This explicitly engages System 2 by enabling systematic problem decomposition and deliberate thought processes.

Abstract vs. Concrete Learning. Research in transfer learning has shown that abstract representations enable superior generalization compared to learning from concrete, surface-level features (Barnett and Ceci, 2002). Our “thought cards” are designed to capture these abstract, generalizable reasoning structures that can be transferred across a wide range of problems. In contrast, raw examples contain domain-specific details that may lead the model to learn spurious, non-generalizable patterns. This distinction explains our method’s superior out-of-distribution performance and the dramatic improvements observed in smaller models (e.g., boosting Llama3-8B from 17.8% to 46.6% on the MATH dataset).

When Pattern-Based Methods Excel. Building on the analysis above, pattern-based methods tend to perform better under specific conditions. They offer significant advantages in complex tasks that can be decomposed into a series of explicit, multi-step thought patterns. Furthermore, the benefits of this structured, pattern-based approach become even more pronounced when the base model has limited intrinsic capabilities, as the explicit guidance provides essential scaffolding for effective problem-solving.

A.4 Failure mode analysis of pattern-based approach

While pattern-based approaches demonstrate strong performance in structured reasoning, it is crucial to analyze their limitations. Understanding these failure modes helps to identify the specific conditions under which traditional example-based

In-Context Learning (ICL) remains a more appropriate choice.

Limitations of Pattern-Based Methods. The primary limitation of pattern-based approaches arises in tasks that are not easily broken down into explicit, repeatable reasoning structures. This includes:

Open-Ended Generation Tasks. Creative endeavors such as writing essays or composing poetry do not follow a rigid logical flow. These tasks rely on creativity, nuance, and a holistic understanding that cannot be easily decomposed into predefined patterns.

Ultra-Low-Frequency Problems For novel or extremely rare problem types, it can be difficult to find and match an appropriate thought pattern from the existing knowledge base. In such "long-tail" scenarios, the lack of a relevant template may hinder performance.

When Example-Based ICL is More Suitable. Conversely, example-based ICL is often more effective and efficient in the following scenarios:

Simple and Solvable Tasks. For straightforward problems where the model already possesses the necessary capabilities to find the answer, providing a few direct examples is a highly efficient method for prompting the correct behavior without the overhead of a complex pattern-matching system.

Tasks Requiring Stylistic Imitation. Many tasks involve learning and replicating a specific style, tone, or format that is difficult to quantify or model as an explicit pattern. Examples include generating literary texts in the style of a particular author or drafting professional emails with a certain etiquette. In these cases, models can implicitly learn the desired stylistic nuances from examples more effectively than from a structured template.

A.5 Human Interpretability

How interpretable are thought patterns to human experts? The thought patterns generated by our method are designed to be highly interpretable to human experts. This interpretability is further enhanced when an abstract pattern is applied to a concrete problem, as it becomes instantiated into a

specific, tangible reasoning process that is easy to follow and verify. We provide detailed examples of these instantiated processes in Figures 14-18.

Can domain experts validate constructed cards?

A key advantage of this high interpretability is that it empowers domain experts to validate and refine the constructed "thought cards." For complex or domain-specific problems, experts can augment the existing knowledge base by improving current cards, designing entirely new ones, or introducing novel, domain-specific reasoning operations. Our automated card construction framework is designed to seamlessly incorporate such expert-driven extensions, facilitating a collaborative human-AI problem-solving process.

A.6 Rationale for Employing Monte Carlo Tree Search

The selection of Monte Carlo Tree Search (MCTS) for constructing thought patterns was a deliberate choice made after considering reasoning task-specific requirements and its comparison with alternative search strategies.

Requirements of Reasoning Tasks. The reasoning tasks often involve constructing complex reasoning chains from a vast combinatorial action space. This setting demands an algorithm that can efficiently search and optimize within such a large space. MCTS is uniquely suited to the sequential and logical nature of reasoning scenarios, which can effectively balance complex trade-offs where other methods fall short.

Comparison with Classical Search Algorithms. Finding optimal reasoning paths requires navigating an exponentially large search space. With an action space of 5 atomic actions and sequence lengths typically up to 8 steps, the number of possible paths approaches $5^8 \approx 400K$. This vast scale renders traditional graph traversal algorithms impractical for our use case.

Depth-First Search (DFS) is prone to premature commitment to suboptimal paths. For complex, multi-step reasoning problems, an initial flawed step can lead the search down an unproductive path from which recovery is difficult, exhibiting a form of brittleness.

Breadth-First Search (BFS) while more exhaustive, typically suffers from combinatorial explosion. By exploring all possible

action combinations simultaneously, it becomes computationally intractable for the vast search space of thought pattern exploration.

MCTS adeptly avoids both pitfalls by using UCT-guided selection to balance exploration of new paths with exploitation of promising ones, thereby identifying effective patterns efficiently. Moreover, we should note that under unlimited computational resources with complete enumeration, DFS and BFS can theoretically guarantee finding the global optimum through exhaustive search and ultimately outperform MCTS. However, in our practical scenario, each action search requires invoking the LLM and depends on the results of previous steps, making DFS/BFS computationally expensive. For example, for a seed sample with a reasoning action sequence of length 5, complete enumeration would require $(5+5^2+5^3+5^4+5^5=3905)$ LLM calls, resulting in unacceptable computational costs. Therefore, in our implementation, we compare MCTS with DFS/BFS under equivalent computational budgets. MCTS achieves a dynamic balance between exploration and exploitation, more efficiently discovering potential optimal solutions under limited resources. In contrast, DFS/BFS often yield sub-optimal solutions under the same budget due to insufficient search space traversal.

Comparison with Evolutionary and Genetic Algorithms. We also explored alternative heuristic methods, including evolutionary algorithms (EAs) such as genetic algorithms (GAs). While these methods can generate diverse thought patterns, they face significant challenges in this domain:

Computational Inefficiency EAs require maintaining large populations of candidate solutions and evaluating them across multiple generations, leading to prohibitive computational costs.

Sequence Coherence Challenges Reasoning chains demand strict logical coherence between sequential steps. Genetic operations like crossover and mutation can easily disrupt this logical flow, producing semantically inconsistent or invalid action sequences.

Fundamental Task Mismatch EAs are highly effective for tasks like prompt search, which

involves optimizing over a large and uncertain vocabulary space. Our task, thought pattern discovery, operates on a relatively small and deterministic set of reasoning actions. For this, the structured, goal-directed search of MCTS is more efficient.

Key Advantages of MCTS for thought pattern Discovery. Given the limitations of other methods, MCTS stands out as the most robust and efficient framework for our objectives. Its primary strengths include:

Natural Structural Alignment. The tree structure of MCTS naturally represents the sequential nature of a reasoning process, where each node corresponds to a distinct reasoning state.

Efficient Pruning. Through intelligent node selection (UCT) and early termination strategies, MCTS significantly reduces the computational overhead by pruning unpromising branches while maintaining the quality of the search.

In summary, MCTS provides the necessary flexibility and efficiency for discovering high-quality, logically coherent reasoning paths, making it the superior choice for our framework.

B Preliminaries

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

B.1 Overall Notations

The definitions for notations are in Table 6.

B.2 LLM Reasoning

LLMs have shown impressive performance across various reasoning tasks (Zhao et al., 2023), including mathematical, commonsense reasoning, and agentic reasoning. Appropriate reasoning methods can substantially enhance LLM problem-solving capabilities, potentially transforming small models into powerful problem-solvers. Given a policy model, π_θ (a pre-trained LLM) and an input problem q , π can autogressive generate an output sequence $output = (s_0, s_1, s_2, \dots, s_T)$ by predicting the next token, where $s_0 := q$ and $ans_p = s_T$. Each output sequence $(s_0, s_1, s_2, \dots, s_T)$ is termed

Table 6: Notation Table in this paper.

Character	Meaning
π_θ	policy LLM
τ_D	specific task
D	demonstration examples of τ_D in in-context learning
q	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 $(s_1, s_2, \dots, s_{t-1})$
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
\mathbb{Q}	seed data
\mathbb{Q}_t	test data

a reasoning trajectory $traj$. The conditional probability distribution of generating the complete reasoning trajectory is:

$$\pi(traj | q) = \prod_{t=1}^T \pi(s_t | q, s_{<t}) \quad (7)$$

Following prior works (Hao et al., 2023; Qi et al., 2025), 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 $(s_0, s_1, s_2, \dots, s_{t-1})$. 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:

$$P_\pi(y_p = y_g | Q) = \mathbb{E}_{(s_0, s_1, \dots, s_T) \sim P_\pi(traj|q)} [P(y_p = y_g | s_0, s_1, \dots, s_T, q)] \quad (8)$$

where $P(y_p = y_g | s_0, s_1, \dots, s_T, q)$ represents the probability of obtaining an accurate answer given the problem q and reasoning trajectory $traj$.

B.3 In-Context Learning

Originally introduced by previous study (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., 2024b). 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:

$$\begin{aligned} D &\sim P(X, Y), \\ y_p &= \pi_\theta(D, x) \end{aligned} \quad (9)$$

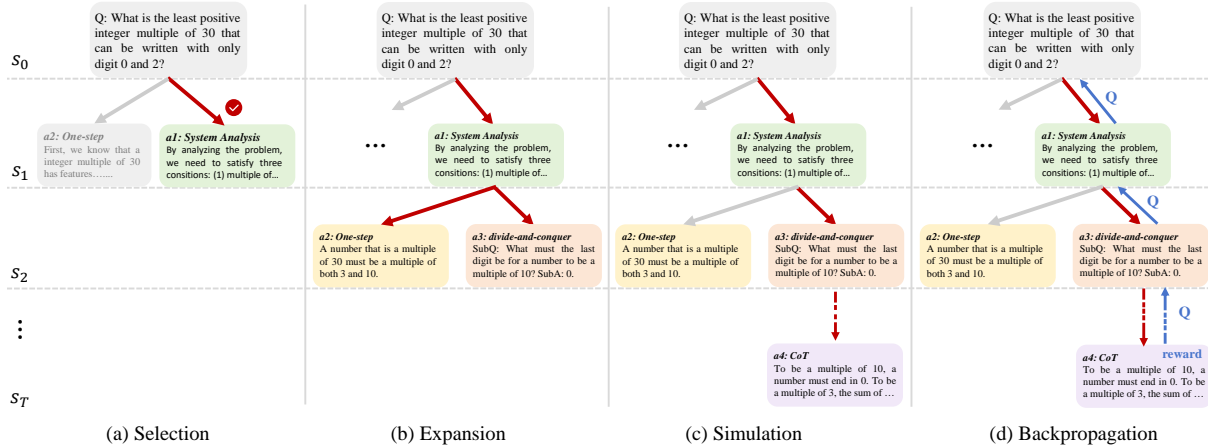


Figure 7: An illustration of four phases in an iteration of MCTS for complex reasoning tasks (B.4).

Extensive research has focused on constructing high-quality examples and enriching the demonstration space D (Luo et al., 2024; Zhou et al., 2024b). For example, Chain-of-Thought (CoT) reasoning (Wei 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. Self-Consistency (Wang et al., 2023b) further enhances performance by generating multiple reasoning paths and selecting the most consistent answer. Prompt Space (Shi et al., 2024) optimizes prompt engineering for better example-based analogy. Many-Shot ICL (Agarwal et al., 2024) enhances performance by scaling sample quantity, which requires substantial context-processing capabilities and computational overhead. Increasing sample volume may even degrade performance by overwhelming model capacity (Liu et al., 2024).

However, traditional ICL paradigm primarily emphasizes example-level analogical learning, with performance constrained by the selection of demonstrations. 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 thought patterns, expanding the concept of context. This shift improves generalization, enabling fully automated, efficient inference without human intervention, even for models with fewer than 10B parameters. The most relevant prior work is Buffer-of-Thought (BoT) (Yang et al., 2024c), which designs task-specific reasoning templates and retrieves relevant ones based on problem

semantics to guide LLM reasoning. However, BoT has some potential limitations, and ThoughtICR differs conceptually in three aspects: (1) *Template Design*: BoT uses monolithic templates with one-to-one problem mapping, which limits flexibility. In contrast, ThoughtICR employs action-driven hierarchical templates composed of decomposable atomic action units, enabling modular reasoning and greater adaptability for complex problems; (2) *Template Construction*: BoT relies on task-specific templates extracted by powerful LLMs (e.g., GPT-4), while ThoughtICR automatically constructs task-agnostic thought cards through MCTS based on atomic reasoning actions. Whether BoT applies to weaker models like Llama3-8B still remains a challenge; (3) *Template Utilization*: BoT retrieves templates via semantic similarity tailored to specific tasks, while ThoughtICR uses a comprehensive complexity framework (CC, SC, PS) that more accurately reflects intrinsic problem characteristics. These differences enable ThoughtICR to achieve impressive cross-domain generalization, reduced dependence on model capabilities, significantly improved computational efficiency, and more flexible practical applications. We also provide direct experimental comparisons in Table 22.

B.4 Monte Carlo Tree Search

As a heuristic search algorithm, MCTS has demonstrated remarkable success in complex reasoning and decision-making environments (Zhou et al., 2024a). 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. 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 7 illustrates the four phases in an iteration, expanding the tree and then updating reward values.

Leveraging MCTS, recent approaches like rStar (Qi et al., 2025) 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 thought patterns (referred to as "thought cards" in Sec. 3.2) and references these thought cards during inference to achieve efficient reasoning. This design enables ThoughtICR 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.

B.5 Verification Methods

Selecting the most accurate reasoning trajectory among multiple candidate solutions presents a critical challenge. This paper explores a simple yet effective method: Self-Consistency (SC) 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:

$$traj_{rm} = \arg \max_{traj} \sum_{i=1}^N \mathbb{I}(y_i = y) \quad (10)$$

PRM PRM ($\mathcal{D} \times \mathcal{S} \rightarrow \mathbb{R}^+$) aims to score each step s_t in the reasoning trajectory:

$$\mathcal{L}_{PRM} = \sum_{t=1}^T y_{s_t} \log(r_{s_t}) + (1 - y_{s_t}) \log(1 - r_{s_t}) \quad (11)$$

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, prior work (Lightman et al., 2024) frames PRM training as a three-class classification task, where human annotations categorize steps as ‘good’, ‘neutral’, or ‘bad’. 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 (Wang et al., 2024b), we utilize the minimum score across all steps to represent the final score of a solution s :

$$traj_{rm} = \arg \max_{traj} \sum_{i=1}^N \mathbb{I}(y_i = y) \cdot rm(p, traj_i) \quad (12)$$

where $rm(p, traj_i)$ 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 .

In this paper, we introduce a simple yet effective two-stage verification method. First, we apply process-supervision scoring to evaluate each reasoning path. The top-3 highest-scoring paths then undergo self-consistency checks to determine the final solution. Our experiments in Table 5 confirm that even these straightforward self-consistency checks effectively identify precise reasoning chains without additional supervision. **Note that, this process does not leak any heuristics towards specific tasks.** The verification mechanism operates independently from the problem-solving process, maintaining task-agnosticity. The PRM and self-consistency check evaluate solution quality based on general reasoning principles rather than task-specific shortcuts.

C Algorithm Details

Overview As shown in Figure 2, ThoughtICR consists of two components: 1) MCTS-powered thought card construction (offline); and 2) adaptive reasoning and verification (online). The algorithm is detailed in Algorithm 1.

C.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.,

Table 7: Comparison with other tree-based search methods. Note that, most methods 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) ReST-MCTS* (Zhang et al., 2024a)	a_2 : one-step thought
RAP (Hao et al., 2023) BEATS (Sun et al., 2024)	a_4 : divide and conquer a_1 : system analysis, a_2 : one-step thought, a_5 : self-refinement
MCTSr (Zhang et al., 2024b) LLaMA-Berry (Zhang et al., 2025)	a_3 : chain-of-thought, a_5 : self refinement
Ours	a_1, a_2, a_3, a_4, a_5

2024a). As illustrated in Qi et al. (2025), existing approaches typically involve a restricted action space (Table 7), 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.
- (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.

C.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 prior work (Qi et al., 2025), 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 (Eq. 1).

C.3 Thought Card Visualization

For better visualization, we provide a detailed example thought card in Figure 8. Note that, We ensure thought card quality in a systematic way: (1) *Correctness*: During thought card construction, each problem pairs with its ground truth answer. The abstract thought on each card represents correct optimal paths for specific problems. This ensures that our thought cards capture thought patterns that lead to correct solutions; (2) *Quality*: We select reasoning paths using a VOC-inspired metric (Equation 3), which balances effectiveness (reward) with computational cost (sequence length). This ensures concise, high-quality thought patterns while avoiding redundancy.

C.4 Card Matching

Card Matching Metrics. Within our framework, we introduce three established metrics to guide card matching: (1) Condition Complexity (CC), measuring the number of distinctive problem conditions; (2) Subquestion Count (SC), which quantifies the number of decomposable subproblems; and (3) Problem Semantic (PS), which evaluates the semantic embeddings of a question’s linguistic features. Note that, we utilize an LLM (e.g., Llama3-8B) to compute CC and SC, and a semantic

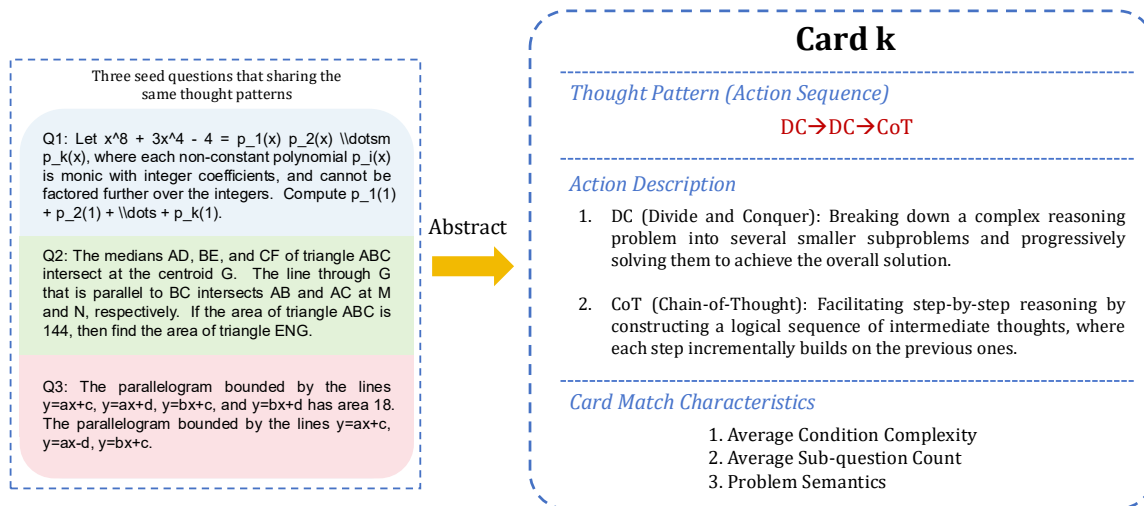


Figure 8: Thought Card Visualization. On the left, the seed dataset contains 3 questions with the same thought pattern “DC→DC→CoT”, corresponding to card k on the right. The card includes both the thought pattern and characteristics used for matching during testing, based on cognitive complexity metrics. When a test question q matches card k , reasoning follows the corresponding pattern “DC→DC→CoT”.

embedding model BAAI/bge-m3¹ for PS computation. Our analysis in Section 4.6 (Table 5) indicates that CC holds a distinct advantage over other individual metrics. Furthermore, as shown in Table 24, it remains competitive even against more complex joint metrics. Consequently, we adopt the simple yet effective CC as our primary metric for its strong balance of performance and simplicity.

CC Rationale. The effectiveness of CC stems from its ability to capture a problem’s **implicit logical structure rather than its surface-level information**. Unlike a simple statistical measure, CC estimates the “prior information load” required for a solution by assessing the extraction and integration of implicit conditions and theorems. This allows it to adeptly handle diverse problem types. As shown in Table 8, lengthy but logically simple problems (e.g., arithmetic word problems) often contain redundant information and thus receive a low CC score. Conversely, concise but complex problems (e.g., mathematical challenges) that require extensive knowledge reconstruction are assigned a high CC score, making it a robust indicator of true complexity. To ensure its stability in practice, our implementation prompts the model to generate three distinct partitioning strategies for a given problem, using the average of these partitions as the final CC value.

¹BAAI/bge-m3

C.5 Verification

For verification, we utilize off-the-shelf PRM: Llama3.1-8B-PRM-Mistral-Data². By default, our experimental results are reported based on PRM+SC verification. Moreover, as demonstrated in Table 5 in Section 4.6, 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.

D Detailed Experimental Setups

D.1 Models

ThoughtICR is a general approach applicable to various LLMs. In our experiments, we evaluate its effectiveness across: 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), DeepSeek-R1-Distill-Qwen-1.5B (Guo et al., 2025), and GPT-4o. Unless stated otherwise, all models are the Instruct version. In Appendix E.5, we analyze the performance of ThoughtICR combined with base models. By focusing on LLMs with parameters generally under 10B, we aim to demonstrate the robustness and efficiency of our method. We expect that applying ThoughtICR to

²RLHFlow/Llama3.1-8B-PRM-Mistral-Data

Table 8: Illustrative examples of the Condition Complexity (CC) metric, contrasting a lengthy but simple problem with a concise but complex one.

Example 1: Long question with low CC (CC = 2)	Example 2: Short question with high CC (CC = 4)
<p>Original Question: 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?</p> <p>Rephrased Question:</p> <ul style="list-style-type: none"> • Condition 1: Initially, there are 15 trees in the grove. • Condition 2: After Grove workers plant additional trees, the total number of trees in the grove will increase to 21. [cite: 1] • Question: How many trees did the grove workers plant today? 	<p>Original Question: Find the sum of all integer bases $b > 9$ for which 17_b is a divisor of 97_b.</p> <p>Rephrased Question:</p> <ul style="list-style-type: none"> • Condition 1: Let b be an integer base such that $b > 9$. • Condition 2: 17_b represents the number $1 \times b + 7$ in base 10, and 97_b represents the number $9 \times b + 7$ in base 10. • Condition 3: Number a divides c if there exists integer k such that $c = a \times k$. • Condition 4: Find all $b > 9$ where 17_b divides 97_b, then sum these bases. • Question: What is the sum of all integer bases $b > 9$ such that 17_b divides 97_b?

small language models will achieve results comparable to or exceeding closed-source LLMs.

D.2 Datasets

The datasets utilized in this paper are listed in Table 9. Note that, for GSM8K, MATH, and STG, we use 200 randomly sampled training instances as seed data and evaluate on their respective test sets for better analysis in cross-domain generalization experiments. For other benchmarks, we reuse 200 MATH samples as seed data to demonstrate the flexibility and generalization of our thought cards and the reasoning paradigm.

- **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 testing set of elementary-level mathematical word problems (MWPs). 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., 2025; Zhang et al., 2024a), we utilize MATH500 as our test set for a fair comparison, a representative and highly challenging 10% subset of MATH.
- **AMC (Li et al., 2025):** This dataset contains problems from the American Mathematics Competitions, specifically focusing on high-school level mathematical challenges. The AMC problems typically require sophisticated mathematical reasoning and creative problem-solving strategies beyond basic arithmetic. For our evaluation, we use a subset of 40 carefully selected problems that demonstrate diverse mathematical concepts and varying complexity levels.
- **AIME25 (Li et al., 2025):** This dataset features problems from the American Invitational

Table 9: Detailed information on the datasets and test set sizes used in our experiments.

Category	Dataset	#Test Samples
Mathematical Reasoning	MATH (Hendrycks et al., 2021)	500
	AMC (Li et al., 2025)	40
	AIME25 (Li et al., 2025)	30
Arithmetic Reasoning	GSM8K (Cobbe et al., 2021)	1319
	SVAMP (Patel et al., 2021)	300
Graduate-level Scientific Reasoning	GPQA-Diamond (GPQA-D) (Rein et al., 2024)	198
General Knowledge Reasoning	MMLU-ProX (MMLU-P) (Xuan et al., 2025)	1351
Logical Reasoning	LogiQA-v2 (Liu et al., 2023)	1572
Commonsense Reasoning	StrategyQA (STG) (Geva et al., 2021)	687
Agentic Reasoning	BALROG (Paglieri et al., 2025)	500

Mathematics Examination, representing some of the most challenging high-school mathematics competitions. AIME problems require advanced mathematical techniques and sophisticated problem-solving approaches, often involving number theory, combinatorics, and geometric reasoning. We evaluate on 30 representative problems that showcase diverse mathematical domains.

- **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. This dataset challenges models to perform multi-hop reasoning across diverse domains of commonsense knowledge.
- **GPQA-Diamond (Rein et al., 2024)**: GPQA (Graduate-level Physics Questions and Answers) is a PhD-level scientific knowledge reasoning benchmark that contains complex questions requiring advanced domain expertise. The Diamond subset consists of 198 particularly challenging questions that test deep scientific knowledge and sophisticated reasoning capabilities. These questions often require integration of multiple scientific concepts and principles to reach the correct solution.
- **MMLU-ProX (Xuan et al., 2025)**: This dataset extends the Massive Multitask Language Understanding (MMLU) benchmark with more challenging, reasoning-intensive questions across diverse academic disciplines. The ProX vari-

ant focuses on complex multi-step reasoning problems that require deeper understanding beyond simple factual recall. We evaluate on 1351 English samples spanning various academic domains, such as engineering, law, biology, and business.

- **LogiQA-v2 (Liu et al., 2023)**: This dataset contains logical reasoning problems that require systematic deductive and inductive reasoning skills. The questions test various forms of logical reasoning including conditional reasoning, categorical syllogisms, and logical puzzles. We use 1572 test samples that represent diverse logical thought patterns and complexity levels.
- **BALROG (Paglieri et al., 2025)**: This benchmark evaluates agentic reasoning capabilities through complex, multi-step decision-making scenarios. Problems require models to reason about agent behaviors, plan sequences of actions, and understand causal relationships in dynamic environments. We evaluate on the classic babyai game (500 samples) that test sophisticated agentic reasoning skills.

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 of the approach.

Table 10: Statistics of reasoning complexity and action chains across datasets.

Dataset	Difficulty	Avg. Length of Action Chain	Number of Action Chains
GSM8K	*	3.09	35
MATH	**	3.59	42
AMC	***	3.67	45

Table 11: 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 settings
exploration weight w	2.0	
predefined terminal threshold c	0.90	
balance factor k	0.95	VOC-based optimal path selection

Table 12: Comparison with leading LLMs. The best results in each box are highlighted in **bold**. Results for all baseline models are sourced from corresponding official websites when accessible. ThoughtICR (Qwen2.5-7B-Instruct) outperforms larger models and powerful closed-source models.

Model	Size	MATH	GSM8K	Average
<i>Closed-Source Models</i>				
GPT-4o-0806 (OpenAI, 2024)	-	77.2	96.1	86.6
GPT-4o mini-0718 (OpenAI, 2024)	-	70.2	93.2	81.7
Claude-3.5-Sonnet-0620 (Anthropic, 2024)	-	71.1	96.4	83.8
Gemini-1.5-Pro (Google DeepMind, 2024)	-	67.7	90.8	79.3
<i>Open-Source Models (>30B)</i>				
Llama3.1-405B-Instruct (Meta AI, 2024)	405B	73.8	96.8	85.3
Qwen2-72B-Instruct (Yang et al., 2024a)	72B	69.0	93.2	81.1
Llama3.3-70B-Instruct (Meta AI, 2024)	70B	77.0	-	77.0
Llama3.1-70B-Instruct (Meta AI, 2024)	70B	68.0	95.1	81.6
Llama3-70B-Instruct (Dubey et al., 2024)	70B	50.4	93.0	71.7
Yi-1.5-34B-Chat (Young et al., 2024)	34B	50.1	90.2	70.2
<i>Ours ($\leq 14B$)</i>				
Qwen2.5-14B-instruct (Qwen Team, 2024)	14B	81.4	95.8	88.6
Qwen2.5-7B-instruct (Qwen Team, 2024)	7B	80.6	93.7	87.2
Qwen2-7B-instruct (Yang et al., 2024a)	7B	66.8	91.8	79.3
Yi-1.5-6B-Chat (Young et al., 2024)	6B	57.4	86.4	71.9
Llama3-8B-Instruct (Dubey et al., 2024)	8B	46.6	89.6	68.1
Llama3.1-8B-Instruct (Meta AI, 2024)	8B	58.0	90.7	74.4
DeepSeek-R1-Distill-Qwen-1.5B (Guo et al., 2025)	1.5B	49.0	88.3	68.7

D.3 Card Distribution

We analyze the length and diversity of action chains across datasets of varying difficulty. As shown in Table 10, both metrics increase with problem complexity. However, many thought patterns, such as “decompose and think step by step” remain con-

sistent across datasets. This consistency helps explain the strong out-of-domain generalization of thought cards, as demonstrated in Figure 5. Additionally, not all five actions are required for each problem. Simpler problems, like Q1 in Figure 2, may require only one action (*CoT*), whereas more complex problems, like Q4, involve longer chains

($SA \rightarrow OST \rightarrow SRR \rightarrow CoT$). The number and complexity of actions thus depend heavily on the specific problem at hand.

D.4 Baselines

For baselines compared in this paper, we conduct experiments on 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 when accessible. In cases where reported results are unavailable but open-source code is provided, we reproduce experiments following the official official settings.

D.5 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 LLM to explicitly state its solution following a pre-defined 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.

D.6 Implementation Details

Setup. Based on the vLLM framework³, all experiments are conducted on a Ubuntu 22.04 machine equipped with an NVIDIA A100-80GB GPU. We list hyperparameters in Table 11.

Seed Data Construction. For GSM8K, MATH, and STG, we use 200 randomly sampled training instances as seed data and evaluate on their respective test sets for better analysis in cross-domain generalization experiments. For other benchmarks, we reuse 200 MATH samples as seed data to demonstrate the flexibility and generalization of our thought cards and the reasoning paradigm.

Pattern Match Metrics. Considering both simplicity and experimental results in Table 5 and Table 24, we use CC as the default pattern matching metric in this paper. We have discussed the detailed rationale in Appendix C.4, and more advanced settings are left for future work. Note that, we use

³<https://github.com/vllm-project/vllm>

BAAI/bge-m3 for PS computation, and an LLM like Llama3-8B for CC/SC computation. Details are provided in Appendix C.4.

Verification. For the verification module, we utilize the off-the-shelf RLHFlow/Llama3.1-8B-PRM-Mistral-Data. More detailed descriptions are illustrated in Appendix C.5.

E Supplementary Results

This section presents supplementary results and analysis, including: comparison with powerful LLMs and baseline methods (E.1), combination with large reasoning models (E.2), weak-to-strong generalization (E.3), multi-task system (E.4), integration with SFT (E.5), performance across problem complexity (E.6), detailed computational cost (E.7), Effective and Efficient TTS (E.8), additional ablation study (E.9), thought pattern analysis (E.10), hyperparameter sensitivity (E.11), and statistical validation (E.12).

E.1 Detailed Comparison with Powerful LLMs and Baseline Methods

Table 12 presents a performance comparison between our method and leading open-source and closed-source models. By employing ThoughtICR, both our 14B and 7B Qwen2.5 models achieve superior results, surpassing many powerful models with over 100B parameters. Notably, on the challenging MATH dataset, our method demonstrates a significant performance advantage, underscoring its effectiveness in complex reasoning tasks. This highlights the strength of our approach in achieving robust reasoning capabilities while utilizing relatively small models. This experiment aims to show how our method enables open-source models to match larger, more powerful counterparts. We consider this a significant contribution to the community, as it provides an effective pathway to democratize high-level reasoning without relying on massive-scale models.

We have extended Table 1 to advanced ICL methods (e.g. reinforced ICL), and tree-based methods (e.g. ToT) for a more comprehensive comparison in Table 13. Due to some lacking results in the original papers, we only present results on widely reported MATH, GSM8K, SVAMP, and StrategyQA. As shown in Table 13, ThoughtICR consistently outperforms advanced ICL methods and tree-based methods. For example, Llama3-8B’s

Table 13: Comparison of ThoughtICR’s reasoning performance with tree-based methods and advanced ICL methods across four benchmarks. “Reinforced ICL” refers to the advanced ICL technique from Many-Shot In-Context Learning (Agarwal et al., 2024), evaluated with 10-shot examples. “CoT+AS” denotes directly providing a high-level action sequence in the instruction, and “CoT+Dynamic” uses the original question-path pairs from the five retrieved thought cards as dynamic examples. ToT and ReST-MCTS* serve as representative baselines from tree-based methods. The best results in each box are highlighted in **bold**. All models are instruct versions.

Model	Method	Mathematics	Arithmetic		Commonsense	Average
		MATH	GSM8K	SVAMP	StrategyQA	
GPT-4o	CoT+AS	80.8	95.5	94.0	80.8	87.8
	CoT+Dynamic	81.8	95.5	94.0	78.4	87.4
	Reinforced ICL	81.0	95.5	94.0	77.7	87.0
	ToT	75.8	95.2	92.7	75.5	84.8
	ReST-MCTS*	78.0	94.6	93.3	76.9	85.7
	Ours	84.8	96.0	94.7	82.2	89.4
Qwen2.5-14B	CoT+AS	78.2	94.3	91.7	72.5	84.2
	CoT+Dynamic	78.0	94.0	93.7	73.3	84.7
	Reinforced ICL	78.8	93.7	93.7	70.3	84.1
	ToT	71.4	93.8	92.0	73.3	82.6
	ReST-MCTS*	74.1	94.0	92.3	71.4	82.9
	Ours	81.4	95.8	93.7	77.3	87.0
Qwen2.5-7B	CoT+AS	77.2	92.7	92.7	71.8	83.6
	CoT+Dynamic	77.0	92.4	92.0	72.0	83.3
	Reinforced ICL	76.6	91.3	92.0	72.5	83.1
	ToT	68.4	91.7	92.1	71.3	80.8
	ReST-MCTS*	72.2	92.4	91.7	69.5	81.5
	Ours	80.6	93.7	93.0	76.0	85.9
Qwen2-7B	CoT+AS	58.2	87.7	89.7	68.5	76.0
	CoT+Dynamic	56.0	88.0	90.7	69.0	75.9
	Reinforced ICL	54.8	87.9	91.7	66.8	75.3
	ToT	53.3	79.0	85.7	66.7	71.1
	ReST-MCTS*	52.4	82.3	86.8	64.9	71.6
	Ours	66.8	91.8	92.7	72.0	80.8
Llama3-8B	CoT+AS	32.6	83.0	90.3	68.1	68.5
	CoT+Dynamic	38.8	83.3	90.3	69.5	70.4
	Reinforced ICL	25.6	81.0	89.0	65.2	65.2
	ToT	13.6	69.0	79.8	60.4	55.7
	ReST-MCTS*	34.2	75.5	88.0	65.0	65.7
	Ours	46.6	89.6	92.7	73.4	75.6

accuracy on MATH improved from 13.6% (ToT) to 46.6% (ThoughtICR), a $3.4\times$ improvement.

We further compare ThoughtICR with other tree-based methods that require extensive training data. As shown in Table 14. Baseline results are directly sourced from their original papers. To evaluate ThoughtICR’s effectiveness on less-optimized models, we specifically test on relatively weaker models including Llama2-7B and Mistral-7B. Despite being training-free with only 0.2K seed samples, ThoughtICR achieves competitive or superior performance. On Llama2-7B, ThoughtICR matches Math-shepherd’s 21.6% accuracy while using $1350\times$ less data (0.2K vs 270K). On Mistral-

7B, ThoughtICR reaches 35.6%, outperforming MCTS-DPO (34.7%) and AutoPSV (22.2%) despite their reliance on 24K and 7.5K training samples. These results demonstrate ThoughtICR’s remarkable data efficiency and robustness across varying model capabilities, making it particularly valuable for resource-constrained scenarios.

E.2 Combination with Large Reasoning Models

Our main results show that ThoughtICR enables smaller models (7B) to approach the performance of larger models like GPT-4o, highlighting its potential to enhance reasoning in small models. As

Table 14: Comparison with other tree-based baselines on MATH. ThoughtICR achieves impressive performance while requiring minimal seed data (0.2K).

Method	Training-free	Prior Data	Llama2-7B	Mistral-7B	Llama3.1-8B
Math-shepherd (Wang et al., 2024c)	✗	270K	21.6	33.0	52.8
RISE (Qu et al., 2024)	✗	7.5K	10.4	18.4	-
MCTS-DPO (Xie et al., 2024)	✗	24K	-	34.7	-
AutoPSV (Lu et al., 2024)	✗	7.5K	-	22.2	52.2
ThoughtICR (Ours)	✓	0.2K	21.6	35.6	58.0

Table 15: Performance of ThoughtICR combined with large reasoning models on the MATH dataset.

Method	MATH Acc (%)
QwQ-32B-Preview	90.6
o1-preview	85.5
ThoughtICR (Qwen2.5-7B)	80.6
ThoughtICR (QwQ-32B-Preview)	93.6

Table 16: Performance of ThoughtICR combined with additional long-CoT reasoning models, including Qwen3 series, DeepSeek-R1, and OpenAI-o1.

Model	Method	AIME25	GPQA-D	Avg.
Qwen3-4B	Base	63.3	55.6	61.5
	+Ours	70.0	60.1	65.0
Qwen3-8B	Base	66.7	61.6	64.2
	+Ours	70.0	65.7	67.9
DeepSeek-R1	Base	70.0	71.7	70.9
	+Ours	76.7	75.8	76.3
OpenAI-o1	Base	80.0	77.8	78.9
	+Ours	83.3	80.8	82.1

shown in Table 15 and 16, it can also be combined with larger reasoning models like QwQ-32B-Preview for even better performance.

E.3 Weak-to-Strong Generalization

As described in prior work (Yang et al., 2024d), interactions between weak and strong models can be categorized into three primary paradigms: 1) weak-to-strong improvement, where models with limited capabilities can effectively guide the development of more advanced models, 2) self-improvement, wherein the weak and strong models are identical, focusing on designing methods to enhance the model’s own performance, and 3) knowledge distillation, which involves transferring the capabilities or knowledge from strong models to weak models. While our previous analysis demonstrated ThoughtICR’s effectiveness in knowledge distilla-

tion and self-improvement, here we focus on its potential for weak-to-strong generalization.

In our experimental setup, we leverage a weaker model to construct thought cards that serve as reasoning guidelines for stronger models, establishing a weak-to-strong generalization framework. Table 17 presents comprehensive results across different model configurations. In weak-to-strong scenarios, our method consistently outperforms CoT baselines, achieving superior performance across all evaluated tasks. Notably, Llama2-7B model can effectively guide GPT-4o, improving performance on MATH (81.6% vs. 77.2%) and GSM8K (95.5% vs. 94.3%). Similar patterns emerge when guiding Qwen2.5-7B and Qwen2.5-14B with Llama3-8B, yielding average improvements of 4.2% and 7.0% respectively. These results demonstrate that structured thought patterns extracted by weaker models can enhance stronger models’ performance, offering a promising direction for scenarios where powerful teacher models are unavailable.

E.4 Multi-Task System

Multi-task learning frameworks represent a promising direction to enhance general system performance. Here, we aim to explore a generalized system capable of simultaneously handling multiple tasks.

Specifically, we extract 50 examples from each of three reasoning domains: mathematical, arithmetic, and commonsense reasoning. These 150 seed data samples are used to construct high-level thought cards, providing guidance signals for subsequent inference. As shown in Table 18, ThoughtICR still achieves competitive performance, further demonstrating the versatility of our reasoning paradigm. This provides new insights for future research on multi-task general systems, suggesting that exploring high-level patterns across tasks could represent a promising direction.

Table 17: Weak-to-strong generalization results. We extract thought patterns using weaker models (∇ left) as guidance for stronger models (∇ right) during inference. Notably, Llama2-7B successfully improves GPT-4o’s performance across evaluated tasks.

Method	MATH	GSM8K	SVAMP	STG	Average	Δ (\uparrow)
<i>Weak-to-Strong (Llama2-7B \rightarrow GPT-4o)</i>						
CoT	77.2	94.3	94.0	79.5	86.2	-
ThoughtICR (Ours)	81.6	95.5	94.3	80.4	88.0	+1.8
<i>Weak-to-Strong (Llama3-8B \rightarrow Qwen2.5-7B)</i>						
CoT	75.5	91.6	92.3	67.6	81.7	-
ThoughtICR (Ours)	80.6	93.7	93.0	76.0	85.9	+4.2
<i>Weak-to-Strong (Llama3-8B \rightarrow Qwen2.5-14B)</i>						
CoT	80.0	94.8	91.3	53.1	79.8	-
ThoughtICR (Ours)	81.0	95.3	93.7	77.3	86.8	+7.0

Table 18: Multi-task performance of Llama3-8B-Instruct across various reasoning tasks. ‘MT’ refers to generating mixed thought cards to guide subsequent inference, while ‘ICL’ represents the best performance with few-shot CoT. We observe that Our approach is also well-suited as a general multi-task reasoning system.

Method	MATH	GSM8K	SVAMP	StrategyQA	Average
ICL	17.8	74.5	81.0	68.4	60.4
ThoughtICR	46.6	89.6	92.7	73.4	75.6
ThoughtICR-MT	44.2	87.4	91.7	71.2	73.7

Table 19: Integration results of ThoughtICR with SFT on Llama3-8B base models. Δ indicates the performance gain of ThoughtICR compared to the best baseline. The results show that ThoughtICR not only significantly enhances the performance of base models but also continues to provide substantial improvements even after large-scale supervised fine-tuning (SFT). This demonstrates the strong compatibility of our method with SFT and its robustness in boosting reasoning performance across multiple tasks.

Model	Setting	MATH	GSM8K	SVAMP	StrategyQA	Average	Δ (\uparrow)
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	-
	ThoughtICR (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	-
	ThoughtICR (Ours)	46.6	89.6	92.7	73.4	75.6	+15.2

Table 20: 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, fewshot CoT+SC, and our method. The best results are highlighted in **bold**. Δ indicates the performance gain of ThoughtICR compared to the best baseline.

Method	1	2	3	4	5	Average	Δ (\uparrow)
<i>Llama3-8B-Instruct</i>							
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	-
ThoughtICR	88.4	75.6	56.2	32.0	16.5	46.6	+17.8
<i>Qwen2.5-7B-Instruct</i>							
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	-
ThoughtICR	97.7	94.5	92.3	81.1	53.7	80.6	+4.2

E.5 Integration with SFT

Integrating our framework with supervised fine-tuning (SFT) presents an opportunity to expand its applicability and scalability, particularly for complex reasoning tasks. Here, we conduct exploratory experiments to assess the compatibility between the two approaches. As shown in Table 19, we evaluate both pretrained checkpoints and instruction-tuned versions of Llama3-8B.

Our results show that, when applied to base models (Llama3-8B), ThoughtICR achieves substantial performance improvements through its structured guidance mechanism, significantly outperforming both zero-shot and few-shot baselines. The performance gains are particularly notable, reaching improvements of up to 26.6% on average across multiple benchmarks. More importantly, when integrated with instruction-tuned models, ThoughtICR continues to deliver remarkable enhancements, with consistent improvements across all evaluated tasks, demonstrating strong synergy between structured reasoning and SFT techniques.

These findings suggest that our reasoning paradigm can be effectively integrated with SFT to enhance model performance. In future work, we plan to explore more sophisticated integration strategies, such as injecting structured long-chain reasoning data (i.e., thought patterns encoded on thought cards) into the model’s inference process through SFT. This could involve techniques inspired by chain-of-preference optimization (CPO) (Zhang et al., 2024c) and TemplateRL (Wu et al., 2025).

E.6 Performance across Problem Complexity

As shown in Table 20, we present the performance of Llama3-8B-Instruct and Qwen2.5-7B-Instruct on Zero-shot CoT, Few-shot CoT+SC, and our method, ThoughtICR, 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 thought patterns, which help LLMs find a clearer solution more quickly.

E.7 Detailed Computational Cost

In this section, we present a more comprehensive analysis of the time cost of ThoughtICR. Additionally, we include an extended comparison with representative tree-based methods to show ThoughtICR’s high efficiency.

For clarity, we first define the time cost. For downstream evaluation tasks, the average time cost per sample is formally defined in Equation 13, where “total offline construction time” represents the time cost of constructing thought cards using 200 seed samples, and “total online inference time” represents the inference time on n downstream samples. Importantly, the offline construction overhead is fixed and independent of downstream task scale. As the number of test samples (n) increases, the amortized offline cost per sample diminishes. Moreover, in practical deployment scenarios, users primarily experience the online inference time, which is relatively short due to our pre-stored high-level thought patterns.

$$t_{\text{avg}} = \frac{\text{total offline construction time}}{n} + \frac{\text{total online inference time}}{n} \quad (13)$$

We conducted experiments on Llama3-8B on three downstream evaluation tasks: STG, GSM8K, MATH. The results are shown in Table 21. ThoughtICR maintains high efficiency while achieving strong accuracy on downstream tasks. For example, on the STG task, it achieves the highest accuracy while using only 5% of the time required by rStar, demonstrating the efficiency of the proposed method. Moreover, While ThoughtICR incurs higher overhead than traditional ICL, it delivers substantial performance gains (e.g., MATH: 28.8%→46.6%), representing an effective test-time scaling method.

E.8 Effective and Efficient TTS

A central concern in TTS evaluation is whether performance improvements arise merely from extended reasoning time rather than from more principled reasoning strategies (Yan et al., 2025). To address this, we conduct comparative experiments that jointly analyze reasoning time and accuracy across multiple representative TTS methods.

As shown in Table 22, ThoughtICR consistently achieves superior accuracy while reducing inference time compared to baselines such as ToT and Meta Prompting. For example, on *Game of*

Table 21: Time cost comparison between ThoughtICR and representative tree-based methods. The best results are highlighted in bold. ‘off’ and ‘on’ denote average offline and online time cost per sample.

Method	STG Acc (%)	STG Cost (↓)	GSM8K Acc (%)	GSM8K Cost (↓)	MATH Acc (%)	MATH Cost (↓)
Zero-shot	57.2	8.8s	68.3	7.4s	5.8	13.5s
Few-shot CoT	68.4	6.6s	74.5	7.0s	17.8	12.1s
CoT+SC@4	66.8	10.3s	80.6	10.8s	28.8	14.5s
ToT	60.4	567.3s	69.0	623.5s	13.6	809.8s
ReST-MCTS*	65.0	503.5s	75.5	677.8s	34.2	467.2s
rStar	71.5	423.8s	91.1	554.6s	42.9	1105.8s
ThoughtICR	73.4	20.2s (off:6.5 + on:13.7)	89.6	25.5s (off:8.2 + on:17.3)	46.6	159.5s (off:50.8 + on:108.7)

Table 22: Comparison of inference time cost and accuracy across representative TTS methods. Lower cost and higher accuracy are better. All baseline results are sourced from the original papers. To ensure fair comparison, we also utilize GPT-4 as the base model following the baseline settings.

Dataset	ToT	Meta Prompting	PAL	BoT	ThoughtICR (Ours)
Game of 24 Time Cost (↓)	8.73	8.47	5.50	5.17	5.15
Game of 24 Accuracy (↑)	74.0	67.0	64.0	82.4	85.0
MGSM Time Cost (↓)	8.34	8.04	4.81	5.00	3.23
MGSM Accuracy (↑)	86.4	84.8	72.0	89.2	94.8

Table 23: Effectiveness of each atomic action in this paper. We evaluate on Llama3-8B-Instruct.

Action Space	Math	AMC	GSM8K	SVAMP	GPQA	StrategyQA	Avg.
$\{a_3\}$ (i.e. CoT)	17.8	7.5	74.5	81.0	22.7	68.4	45.3
$\{a_3, a_4\}$	42.6	20.0	83.5	88.3	30.3	69.9	55.8
$\{a_2, a_3, a_4\}$	43.2	22.5	85.2	89.7	32.8	70.3	57.3
$\{a_1, a_2, a_3, a_4\}$	46.0	27.5	87.3	91.0	36.9	72.6	60.2
All Five Actions (Ours)	46.6	30.0	89.6	92.7	39.4	73.4	62.0

24, ThoughtICR attains 85.0% accuracy with a cost of only 5.15, outperforming ToT (Yao et al., 2023) (74.0% at 8.73) and Meta Prompting (Suzgun and Kalai, 2024) (67.0% at 8.47). Compared to PAL (Gao et al., 2023), ThoughtICR operates within a similar time cost but yields substantially higher accuracy (e.g., 94.8% vs. 72.0% on MGSM). Furthermore, when compared to BoT (Yang et al., 2024c), ThoughtICR still exhibits clear advantages, improving both accuracy and efficiency (e.g., 85.0% vs. 82.4% on *Game of 24* with slightly lower time cost). These results indicate that ThoughtICR’s improvements do not simply stem from longer inference or brute-force scaling. Instead, they arise from its automated thought-level reasoning framework, which leverages structured thought patterns constructed via MCTS to effectively balance efficiency and performance. In this way, ThoughtICR represents an efficient and effective test-time scaling advancement.

E.9 Additional Ablation Study

In addition to the ablation study on modules 1 (thought card construction), and 2 (thought pattern selection, verification) in the main text, we present additional ablation study on atomic actions and card matching metrics.

Effect of Each Atomic Action We conduct a detailed analysis to evaluate the effectiveness of our atomic actions, focusing on the framework’s sensitivity and its generality across different domains. The results of our ablation study, presented in Table 23, highlight two key findings.

First, our framework demonstrates robustness and is **not overly sensitive to the specific combination of atomic actions chosen**. As the table shows, while performance scales with the inclusion of more actions, various subsets consistently and significantly outperform the traditional CoT baseline. This indicates that our framework effectively leverages any given combination of actions, rather than depending on one perfect configuration.

Second, our choice of atomic actions is in-

Table 24: Analysis of joint metric performance with different weighting schemes on Llama3-8B. While some combinations offer marginal gains, the CC-only default remains a simple yet effective choice.

Metric	MATH	GSM8K	Avg.
Equal weights (CC:1/3, SC:1/3, PS:1/3)	47.4	90.2	68.8
CC-heavy (CC:1/2, SC:1/4, PS:1/4)	47.3	90.5	69.9
SC-heavy (CC:1/4, SC:1/2, PS:1/4)	45.5	87.8	66.7
PS-heavy (CC:1/4, SC:1/4, PS:1/2)	45.1	87.0	66.1
CC only (Our default)	46.6	89.6	68.1

Table 25: Dominant Action Sequences and Their Rationale Across Different Benchmarks.

Dataset Domain	Benchmarks	Action Chain	Ratio (%)	Description / Rationale
Arithmetic	(GSM8K, SVAMP)	<i>CoT</i>	23.8	Simple sequential problems enable direct step-by-step reasoning.
Mathematical	(MATH, AMC)	<i>DC → DC → CoT</i>	14.9	Multi-faceted problems benefit from recursive decomposition before deep reasoning.
Commonsense	(StrategyQA)	<i>SA → OST → CoT</i>	16.8	Logic-based tasks require systematic analysis with insight generation.
Scientific	(GPQA)	<i>SA → DC → SRR → CoT</i>	15.2	Knowledge-intensive tasks require comprehensive analysis, problem decomposition with verification.

tentionally **domain-agnostic, not predefined for a particular problem type**. We evaluate our method, ThoughtICR, across mathematics, arithmetic, scientific, and commonsense reasoning tasks using the same set of five actions: System Analysis, One-Step Thought, Chain-of-Thought, Divide and Conquer, and Self-Reflection and Refinement. The consistent performance gains across these diverse fields underscore the generalizability of our approach. For instance, using Llama3-8B, our method boosts accuracy from 28.8% to 46.6% on MATH (mathematics), from 80.6% to 89.6% on GSM8K (arithmetic), and from 66.8% to 73.4% on StrategyQA (commonsense). This demonstrates that the atomic actions capture fundamental, cross-domain reasoning steps.

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 perception and reasoning tasks.

Effect of Card Matching Metrics In addition to our default metric (CC only), we also explored the effectiveness of the joint metric. This metric combines all three cognitive complexity indicators-Condition Complexity (CC), Subquestion Count (SC), and Problem Semantics (PS) to create a more

comprehensive ranking for thought card selection. Specifically, this approach computes the three indicators for an incoming question and then uses a weighted average to rank and select the most relevant cards.

To investigate the optimal weighting strategy, we experimented with several configurations on Llama3-8B. As shown in Table 24, while certain combinations showed marginal improvements on specific datasets (e.g., equal weights on GSM8K), the simpler CC-only metric demonstrated competitive performance across diverse scenarios. Given its effectiveness and simplicity, we selected the CC-only approach as our default strategy. Detailed analysis of CC are provided in Appendix C.4, and more complex settings are left for future work.

E.10 Thought Pattern Analysis

A critical aspect of our framework is understanding which thought patterns are most effective for different problem domains. Based on the statistical analysis of our experiments, we identified the most frequently selected action sequence (thought pattern) for several key benchmarks. Table 25 summarizes the dominant thought patterns on several representative benchmarks, illustrating a clear taxonomy of which strategies are best suited for specific types of tasks.

The above results reveal a strong correspondence

Table 26: Hyperparameter sensitivity analysis. The optimal values chosen for this paper are highlighted in **bold**.

Hyperparameter	Setting	MATH Acc. (%)	StrategyQA Acc. (%)
Balance factor (k)	0.9	49	71
	0.95	52	73
	1.0	51	72
Early termination threshold (c)	0.8	50	72
	0.9	52	73
	1.0	49	69
Number of thought cards (n)	1	40	61
	3	47	70
	5	52	73
	10	54	74

between the nature of a problem and its optimal reasoning structure. For instance, straightforward arithmetic tasks are efficiently handled by a direct Chain-of-Thought (CoT) process. In contrast, more complex domains, such as scientific and mathematical problems, necessitate more elaborate sequences involving Systematic Analysis (SA) and Decomposition (DC) to break down the problem before reaching the solution. This demonstrates our framework’s ability to adaptively select appropriate, high-level thought patterns for diverse challenges.

E.11 Hyperparameter Sensitivity

We conduct a sensitivity analysis on our method’s key hyperparameters. Our model involves three categories of parameters: (1) vLLM inference settings, following common practice; (2) standard Monte Carlo Tree Search (MCTS) parameters (e.g., exploration weight), following prior work; and (3) method-specific hyperparameters that require tuning. For the latter, we perform a grid search using Llama3-8B-Instruct on a dedicated validation set of 100 problems from the MATH and StrategyQA training set, which is disjoint from our 200 seed samples to prevent data leakage. The results of this tuning process for the balance factor (k), early termination threshold (c), and the number of thought cards (n) are summarized in Table 26.

Our analysis reveals that the model’s performance is robust across a range of settings. The **balance factor** (k), which controls the trade-off between reward and cost (as seen in Equation 3), achieves optimal performance at $k = 0.95$, effectively balancing solution quality with reasoning complexity. The **early termination threshold** (c), which determines when MCTS stops based on consistency, is best set to $c = 0.9$ to allow for opti-

mal exploration while avoiding premature termination. Finally, for the **number of thought cards** (n), which controls thought pattern diversity, performance improves significantly when using multiple cards. We selected $n = 5$ as it offers a strong balance between performance and computational cost, noting diminishing returns beyond this point.

E.12 Statistical Validation

To statistically evaluate the differences between reasoning with and without our ThoughtICR paradigm, we apply the nonparametric Wilcoxon signed-rank test (Kotz and Johnson, 1992). This statistical test is specifically designed to compare two related samples or repeated measurements when the data may not follow a normal distribution, making it particularly suitable for our analysis. The Wilcoxon signed-rank test evaluates whether there is a significant difference between paired observations through the following procedure:

- Calculate differences:** For each pair of values X_i and Y_i , compute the difference $D_i = X_i - Y_i$.
- Rank differences:** Take the absolute values $|D_i|$ and rank them from smallest to largest, denoted as R_i . For ties, average ranks are assigned.
- Assign signs to ranks:** For each pair (X_i, Y_i) , assign the sign of D_i to its corresponding rank: $R'_i = \text{sign}(D_i) \cdot R_i$, where $\text{sign}(D_i) = +1$ if $D_i > 0$, -1 if $D_i < 0$, and 0 if $D_i = 0$.
- Calculate rank sums:** Separate the ranks into positive and negative sums: $W^+ = \sum_{D_i > 0} R'_i$ and $W^- = \sum_{D_i < 0} R'_i$.

Table 27: Statistical significance of differences between reasoning with and without ThoughtICR.

Model	Llama3-8B	Qwen2.5-14B	GPT-4o
p-value	1e-2	2e-2	1e-2

- Determine test statistic:** The test statistic W is the smaller of the two sums: $W = \min(W^+, W^-)$.
- Calculate p-value:** The p-value is derived from the distribution of the test statistic W .

In our analysis, we test the null hypothesis of no significant difference (H_0 : difference = 0) against the alternative hypothesis of a significant difference (H_1 : difference \neq 0). We compare model performance across multiple tasks and dataset: (1) **mathematical reasoning**: MATH (Hendrycks et al., 2021), AMC (Li et al., 2025), AIME25 (Li et al., 2025); (2) **arithmetic reasoning**: GSM8K (Cobbe et al., 2021); (3) **graduate-level science**: GPQA-Diamond (GPQA-D) (Rein et al., 2024); (4) **general knowledge reasoning**: MMLU-ProX (MMLU-P) (Xuan et al., 2025)); (5) **logical reasoning**: LogiQA-v2 (Liu et al., 2023); (6) **common-sense reasoning**: StrategyQA (STG) (Geva et al., 2021); (7) **agentic reasoning**: BALROG (Paglieri et al., 2025).

Following common practice, we use a significance level of 0.05 (5e-2). As shown in Table 27, all p-values are below 0.05, leading us to reject the null hypothesis (H_0). These findings offer strong statistical evidence that incorporating our ThoughtICR framework significantly enhances reasoning performance across all tested dimensions.

F Prompts

We reference and adapt prompts designed by other researchers⁴ to construct the input prompts for our five actions in Figure 9-13. To improve readability, we have applied selective formatting, such as bold highlighting, to enhance the presentation. Note that, each prompt has clearly describe the core definition of the action.

G Case Study

Figures 14-18 present detailed problem-solving examples employing our method and in-context

⁴<https://github.com/reasoning-machines/prompt-lib/tree/main> and <https://github.com/zhentingqi/rStar/tree/main/prompts>

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 14, both ICL and GPT-4o (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?

...

<User Input>

Q: given question

Figure 9: Input prompt for action 1: System Analysis (SA).

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 10: Input prompt for action 2: One-Step Thought (OST).

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 11: Input prompt for action 3: Chain-of-Thought (CoT).

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 12: Input prompt for action 4: Divide and Conquer (DC).

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.

Guidelines:

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 13: Input prompt for action 5: Self-Reflection and Refinement (SRR).

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 S be the least positive multiple of 30 that can be written with only the digits 0 and 2. First, S is a multiple of 10, so its units digit must be 0. S 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 S is minimal, we take exactly three 2's and do not have any additional 0's: $S = 2220$.

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 0 (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, 100 is the smallest such number.

The answer is: 100



ThoughtICR 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 2220.

The answer is: 2220



Figure 14: 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 $\triangle DEF$ with a right angle at E, and we know that $\sin D = 0.7$. We need to find the length of DE.

****Step 1: Identify given information****

The triangle has the following properties:

1. $\angle DEF = 90^\circ$ (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 $\angle DEF$, 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:
 $\sin D = \text{opposite} / \text{hypotenuse} = EF / DF$

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 $DF = 10$: $DE^2 + 7^2 = 10^2$, $DE^2 + 49 = 100$, $DE^2 = 100 - 49 = 51$

****Step 5: Solve for DE****

Take the square root of both sides to find DE: $DE = \sqrt{51}$

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



Few-shot CoT response:

Let's think step by step.

1. By definition, $\sin D = EF / DF$, so $EF = DF \cdot \sin D = 7 \cdot 0.7 = 4.9$.

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

The answer is: $\boxed{4.9}$



Thought/CR 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 $\triangle DEF$ is a right triangle with $\angle DEF = 90^\circ$, and $EF = 7$.

2. Given that $\sin D = 0.7$, we recognize that
 $\sin D = \text{opposite} / \text{hypotenuse} = EF / DF = 7 / DF$.

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

3. Now, since $\triangle DEF$ 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: $\boxed{\sqrt{51}}$



Figure 15: 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

####

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:



ThoughtICR 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 $3 \times 60 = 180$ 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:



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

####

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:



ThoughtICR 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:



Figure 17: 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



Thought/CR 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 18: Example 5 about commonsense on the StrategyQA dataset.