Reversal of Thought: Enhancing Large Language Models with Preference-Guided Reverse Reasoning Warm-up

Anonymous ACL submission

Abstract

Large language models (LLMs) have shown remarkable performance in reasoning tasks but face limitations in mathematical and complex logical reasoning. Existing methods to improve LLMs' logical capabilities either involve traceable or verifiable logical sequences that generate more reliable responses by constructing logical structures yet increase computational costs, or introduces rigid logic template rules, reducing flexibility. In this paper, we propose Reversal of Thought (RoT), a plug-and-play and cost-effective reasoning framework designed to enhance the logical reasoning abilities of LLMs during the warm-up phase prior to batch inference. RoT utilizes a *Preference-Guided Reverse Reasoning* warm-up strategy, which integrates logical symbols for pseudocode planning through meta-cognitive mechanisms and pairwise preference self-evaluation to generate task-specific prompts solely through demonstrations, aligning with LLMs' cognitive preferences shaped by RLHF. Through reverse reasoning, we utilize a *Cognitive Preference Manager* to assess knowledge boundaries and further expand LLMs' reasoning capabilities by aggregating solution logic for known tasks and stylistic templates for unknown tasks. Experiments across various tasks demonstrate that RoT surpasses existing baselines in both reasoning accuracy and efficiency. Our code and datasets are available in supplementary materials.

1 Introduction

Large language models (LLMs) like Qwen (Bai et al., 2023), Llama (Dubey et al., 2024), and GPT-4 (Achiam et al., 2023) have demonstrated remarkable performance in various reasoning tasks via single-step prompting with few shots upon scaling model size (Plaat et al., 2024) but remain restricted in mathematical and intricate logical reasoning domains (Arkoudas, 2023; Stechly et al.), which has spurred more effective multi-step Chain-of-Thought (CoT) prompting (Wei et al., 2022) approaches for activating step-by-step logical capabilities. However, LLMs are prone to unfaithfulness, resulting in cascaded intermediate errors (Bao et al., 2024; Yang et al., 2024b).

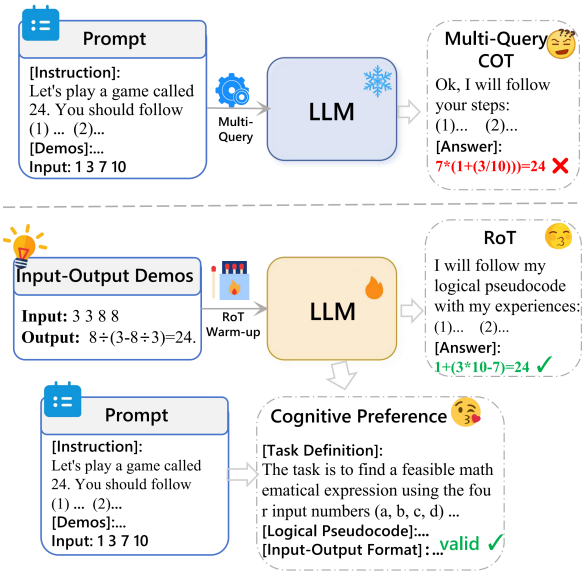

Figure 1: Comparison between CoT (Yao et al., 2024; Besta et al., 2024; Yang et al., 2024a) and Reversal of Thought (RoT)

Recent studies have advanced CoT to guide LLMs, mainly through either multi-step prompting such as introducing planning-and-solve (Plaat et al., 2024; Yang et al., 2024a), self-consistency (Narang et al.; Wang et al., 2024) and recursive reasoning process (Lee and Kim, 2023; Yu et al., 2024) through Tree-of-Thought (ToT) (Yao et al., 2024), Graph-of-Thought (GoT) (Besta et al., 2024), or multi-role (Zhang et al.; Suzgun and Kalai, 2024) to enhance logical capabilities and mitigate hallucination, yet this has stealthily increased inference cost due to the multi-step inference. Buffer-of-Thought (BoT) (Yang et al., 2024a) attempts to reduce thinking steps by leveraging Retrieval-Augmented Gen-

eration (RAG) to retrieve gold thought templates from the buffer. However, it sacrifices flexibility due to the initialization of pre-set manual thought templates. Therefore, achieving accurate reasoning in LLMs while minimizing resource consumption remains a significant challenge.

In summary, existing methods primarily rely on multi-query CoT which injects knowledge (Suzgun and Kalai, 2024; Plaat et al., 2024) or data structure (Yao et al., 2024; Besta et al., 2024) to optimize decisions making, and encounter three significant limitations: **(1) limitation in logical reasoning:** Despite attempting different logic data structures (Yao et al., 2024; Besta et al., 2024; Yang et al., 2024a), an effective initiative Chain-of-Thought paradigm that suits and improves logical reasoning remains elusive (Bao et al., 2024); **(2) unfaithfulness and cascaded errors:** Single-step or multi-step methods are liable to cause LLMs to output hallucinations, leading to cascading logic errors (Bao et al., 2024); **(3) Trade-off between enhanced logic capabilities and resource consumption:** Recent CoT advancements via multi-step or multi-role prompting increase costs and achieving a balance between logical flexibility, accuracy, and cost is of great significance for practical application.

To address above limitations, inspired by meta-cognition (Fleur et al., 2021) and and cognitive preference (Uddin, 2021; Zhou et al., 2023; Margatina et al., 2023), we propose Reversal of Thought (RoT), a plug-and-play and cost-effective framework that enables LLMs to explore cognitive preference on logical pseudocode solely using reverse prompting with given demos without additional task-related affirmations, as depicted in Figure 1. Our key contributions are as follows:

- To the best of our knowledge, we are the first to introduce a reversal reasoning for cognitive preference that enhances logical reasoning in LLMs by combining meta-cognitive with cognitive preference, resulting in a more modular and cost-efficient framework for complex tasks.
- We propose a *Preference-Guided Reverse Reasoning* framework that enhances LLMs' task cognition by employing a reverse reasoning warm-up strategy and preference-based self-evaluation to improve logical reasoning based on LLMs' cognitive preferences.
- We introduce a *Cognitive Preference Manager* to evaluate knowledge boundaries, enabling the automatic adaptation of cognitive preference styles for unknown logic tasks and efficient aggregation of solution logic for known tasks.

2 Related Work

2.1 Chain-of-Thought (CoT) Prompting

Chain-of-Thought (CoT) prompting (Wei et al., 2022) has been proven to be a promising approach that incorporates an intermediate logic chain to enhance LLMs' logic. Recent studies primarily aimed at improving logical accuracy by external validation mechanisms like symbolic reasoning (Cai et al., 2023; Pan et al.), stepwise verification including self-consistency (Narang et al.; Yu et al., 2024; Wang et al., 2024), self-refine (Madaan et al., 2024), self-reflection (Renze and Guven, 2024) and more hierarchical information such as Least-to-Most (Zhou et al.), Cumulative-Reasoning (Zhang et al.) and Multi-experts (Suzgun and Kalai, 2024) strategies, but faced challenges related to cumulative errors (Bao et al., 2024) or poor flexibility (Yang et al., 2024a). Additionally, numerous studies also proposed more standardized recursive or backtracking branch forms from the logical data structure, including Tree-of-Thought (ToT) (Yao et al., 2024), Graph-of-Thought (GoT) (Besta et al., 2024) and Buffer-of-Thought (BoT) (Yang et al., 2024a). However, an efficient logical reasoning method that strikes a balance among reasoning accuracy, flexibility, and cost has yet to be discovered. Our method is activated through meta cognition (Fleur et al., 2021) by introducing reverse reasoning to form effective LLMs-taste prompts within cognitive preference (Uddin, 2021) for plan-and-solve with logical pseudocode at least.

2.2 Knowledge Boundary for Enhancing Large Language Models

Integrating knowledge boundary within LLMs has emerged as a prospective strategy for enhancing their ability to avoid reasoning hallucinations of unknown knowledge through knowledge boundary constraints which requires additional algorithmic efforts (Yin et al., 2024; Chen et al.), external graph knowledge (Tian et al., 2024), and training consumption (Sun et al., 2024). Additionally, they focus on avoiding responses to unknown or incorrect prompts rather than proposing bold and proactive solutions to expand knowledge boundary in a

heuristics without training. We proposed a prompt-based method utilizing LLMs pretrained knowledge boundary, inspired by meta cognition (Fleur et al., 2021) and cognitive preference for unknown knowledge (Uddin, 2021). Our method conducts reverse prompting on probing knowledge through demonstrations to obtain LLMs-taste problem cognitions, aggregates and distills original prompt into cognitive preference version.

3 Reversal of Thought

3.1 Overview

> *Tell me and I forget. Teach me and I remember. Involve me and I learn.*
>
> Franklin (2005)

As the aforementioned wisdom related to human cognitive learning implies, merely telling or teaching is inadequate (Bao et al., 2024). Moreover, most LLMs have undergone extensive pre-training (Achiam et al., 2023; Bai et al., 2023; Dubey et al., 2024) and reinforcement learning from human feedback (RLHF) (Ouyang et al., 2022), instilling in LLMs a propensity for specific cognitive patterns, which manifests in two progressive layers of LLMs-taste description: **(1) Stylistic template:** encompassing grammatical and syntactic structures in descriptions for thinking problems. **(2) Solution logic:** comprising problem-solving reasoning and methodological cues. Therefore, *Reversal of Thought (RoT)* involves answering the following two research questions (RQs):

- ***RQ1:*** *How to make LLMs output preference cognitive templates and logic for specific tasks and activate known cognitive boundaries?*
- ***RQ2:*** *How to autonomously use cognitive templates with incorrect response to expand the possible knowledge boundaries?*

To activate and enhance LLMs logical flexibility, accuracy, and the ability to autonomously construct meta-cognition without training for logical reasoning, inspired by meta-cognition (Fleur et al., 2021) and cognitive preference (Uddin, 2021; Zhou et al., 2023; Margatina et al., 2023), we introduce *Reversal of Thought (RoT)*, a cost-effective paradigm that enables LLMs to first explore cognitive preference on logical pseudocode solely through given examples without additional task-oriented affirmation, activates the pre-trained known logic under *Reverse Reasoning Warm-up* (detailed in section 3.2), and then optimizes the original prompt for LLMs-taste prompt via *Cognitive Preference Manager* (detailed in section 3.3) to determine the transfer of cognitive preference style for unknown logic template and aggregation of known solution logic, as depicted in Figure 2.

Algorithm 1 Preference-Guided Reverse Reasoning (PGRR)

Require: P: Initial prompt, D: Input-output demonstrations, $warm$: Number of warm iterations
(1) Reverse Reasoning Warm-up:
for $i = 1$ **to** $warm$ **do**
  $R^{(i)} \leftarrow \mathcal{M}_{\text{LLM}}(P_r, D, i)$ {Generate candidate responses}
  $P^i_{res} \leftarrow \frac{1}{|R^{(i)}|} \sum_{R_{i,j} \in R^{(i)}} \exp(P(R_{i,j}|P_r, D))$
end for
$R \leftarrow \bigcup_{i=1}^{warm} R^{(i)}$ {Collect all responses}
(2) Pairwise Preference Evaluation:
for $i = 0$ **to** $warm - 1$ **do**
  $P_{pre}(R_{i+1} \succ R_i) \leftarrow \exp(\mathcal{M}_{\text{LLM}}(P_{eval}, R_{i+1}, R_i))$
end for
for $i = 0$ **to** $warm - 1$ **do**
  for $j = 0$ **to** $i - 1$ **do**
    $P_{pre}(R_i \succ R_j) \leftarrow P_{pre}(R_i \succ R_j)$ {Utilize preference transitivity}
  end for
end for
(3) Preference-Guided Ranking:
for $i = 1$ **to** $warm$ **do**
  $\bar{P}_{pre}(R_i) \leftarrow \frac{1}{warm-1} \sum_{\substack{j=1 \\ j \neq i}}^{warm} P_{pre}(R_i \succ R_j)$
end for
$P_{opt} \leftarrow \arg\max_{R_i} \left(\frac{P^i_{res} + \bar{P}_{pre}(R_i)}{2} \right)$
return P_{opt} {Optimal LLMs-taste prompt}

3.2 Reverse Reasoning with Meta-cognition

Preference-Guided Reverse Reasoning. Inspired by RLHF (Ouyang et al., 2022) utilizing preference data, and to derive high-cognitive preference prompt P^* that enhance logical reasoning in LLMs, we propose a *Preference-Guided Reverse Reasoning (PGRR)* framework (detailed in alogrithm 1) mapping input-output demonstrations D from an initial prompt P to an optimal LLM-taste prompt P_{opt} from prompt candidates.

(1) Reverse Reasoning Warm-up. We query the LLM $\mathcal{M}_{\text{LLM}}$ with a reversal prompt and demonstrations $\{P_r, D\}$ (detailed in figure 4) for $warm$ iterations, generating a set of prompt candidates for solution $R = \{R_1, R_2, \ldots, R_{warm}\}$ and their corresponding average probabilities P^i_{res}:

$$R = \bigcup_{i=1}^{\text{warm}} R^{(i)} = \bigcup_{i=1}^{\text{warm}} \mathcal{M}_{\mathcal{LLM}}(P_r, D, i), \quad (1)$$

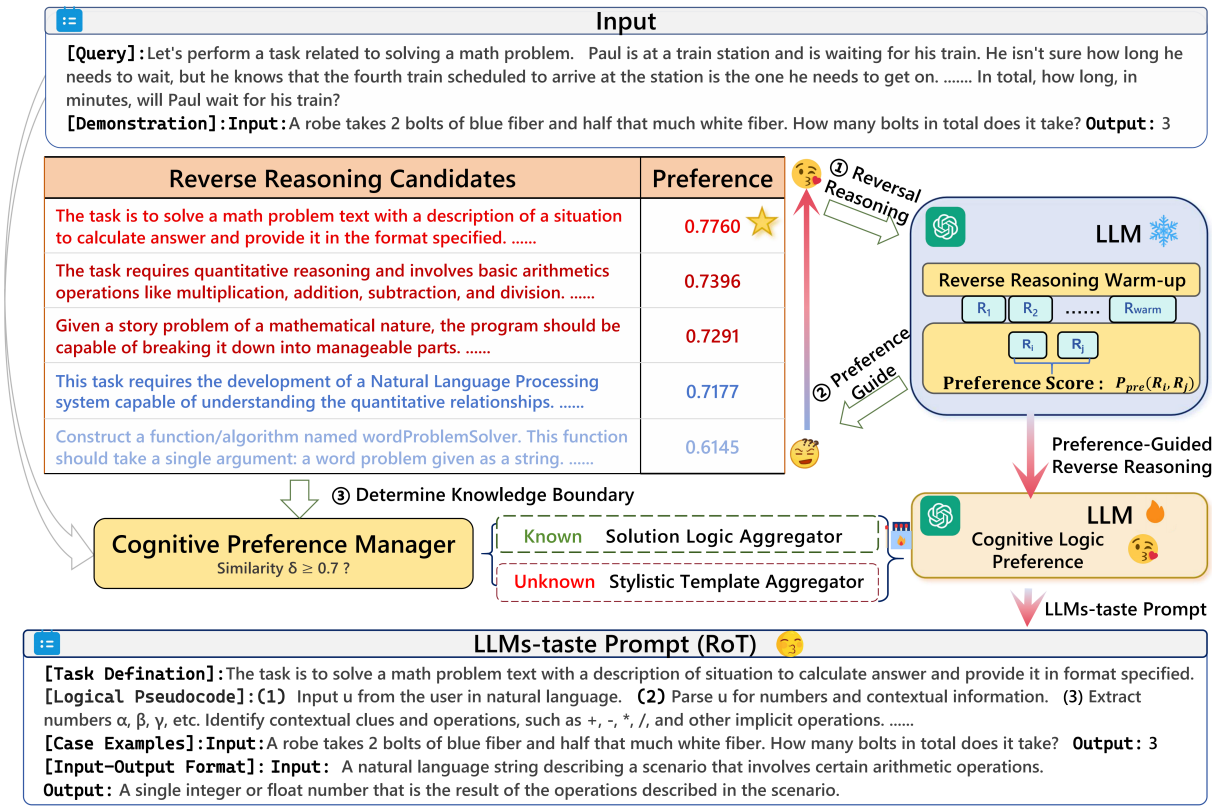

Figure 2: Architecture of Reversal-of-Thought (RoT). RoT comprises two primary components: *Preference Guided Reverse Reasoning*, which enhances logical reasoning by activating LLMs' cognitive preferences, and *Cognitive Preference Manager*, which assesses knowledge boundaries and adapts cognitive styles for various tasks.

$$P^i_{res} = \frac{1}{|R_i|} \sum_{R_{i,j} \in R} \exp(P(R_{ij}|P_r, D)). \quad (2)$$

where $R^{(i)}$ represents the i-th generated response. $\mathcal{M}_{\text{LLM}}(P_r, D, i)$ is the model output based on the reversal prompt P_r and demonstrations D for the i-th iteration. $P(R_{ij}|P_r, D)$ denotes log probability for each token $R_{ij} \in R_i$ from LLMs.

(2) Pairwise Preference Evaluation. To acquire the most LLMs-taste prompt, we pair candidate responses R as data pairs $\{R_i, R_{i+1}\}$ where $i = 0, 1, \cdots, warm - 1$ to calculate the relative preference $P(R_{i+1} \succ R_i)$ through LLM's self-evaluation of its preference for R_{i+1} over R_i, formally define as:

$$P_{pre}(R_{i+1} \succ R_i) = \exp(\mathcal{M}_{\mathcal{LLM}}(P_{eval}, R_{i+1}, R_i)) \quad (3)$$

where $\mathcal{M}_{\mathcal{LLM}}(P_{eval}, R_{i+1}, R_i)$ represents that require LLM to select more preferred data through P_{eval} with a structure as *"Please choose your more preferred instruction (A/B): (A).R_{i+1}; (B).R_i".*

Following the principle of preference transitivity (Liu et al., 2024b), we extend $P(R_{i+1} \succ R_i)$ to $P(R_i \succ R_j)$ to reduce computational cost from, thereby forming a preference matrix $P_{pre} \in \mathbb{R}^{warm \times warm}$, formally:

$$P_{pre}(R_i \succ R_j) = \begin{cases} 1 & i = j \\ \prod_{k=j}^{i-1} P_{pre}(R_{k+1} \succ R_k) & i > j \\ 1 - P_{pre}(R_j \succ R_i) & i < j \end{cases} \quad (4)$$

(3) Preference-Guided Ranking. To identify the most LLMs-preferred and high-quality response, we compute each response R_i's overall preference score $\bar{P}_{pre}(R_i)$, and averaging both average probabilities P^i_{res} in matrix P_{pre} and preference score $\bar{P}_{pre}(R_i)$ to obtain the best LLM-taste prompt P_{opt}:

$$\bar{P}_{pre}(R_i) = \frac{1}{warm - 1} \sum_{\substack{j=1 \\ j \neq i}}^{warm} P_{pre}(R_i \succ R_j), \quad (5)$$

$$P_{opt} = \arg\max_{R_i} \left(\frac{P^i_{res} + \bar{P}_{pre}(R_i)}{2} \right). \quad (6)$$

Reverse Logic for Meta-cognition. Within reverse reasoning, we further follow meta-cognitive (Suzgun and Kalai, 2024) using plan-and-solve by

Algorithm 2 Cognitive Preference Manager (CPM)

Require: P: Original prompt, P^*: Reverse-reasoned prompt, $\mathcal{M}_{emb}$: Offline LLM embedding model, δ: Similarity threshold

$P_{task} \leftarrow P, P^*_{task} \leftarrow P^*$
$s \leftarrow \mathcal{M}_{emb}(P_{task}, P^*_{task})$ Calculate embedding similarity between P_{task} and P^*_{task}.
if $s \geq \delta$ **then**
  Known detected: Enhance and refine P to optimized instructions P_{final}.
  Aggregate relevant task-specific knowledge.
  return P_{final}.
else
  UnKnown detected: Adapt and expand P_{task}.
  Leverage cognitive preference templates T and P to generate optimized instructions P_{final}.
  return P_{final}.
end if

integrating logical algorithm pseudo-code to improve reasoning comprehension. And we incorporate fundamental mathematical logic symbols, including logical operators, quantifiers, inequalities and conditional statements, to facilitate model reasoning detailed in Appendix A.2.

3.3 Cognitive Preference Manager

Cognitive Preference Manager. After reverse reasoning for LLMs-cognitive description P^*, We introduce an offline-deployed LLM embedding model $\mathcal{M}_{emb}$ to assist *Cognitive Preference Manager (CPM)* in determining whether reverse reasoning under reverse prompt P_r and demonstrations D reaches the knowledge boundary or cognitive error by calculating the similarity and setting a threshold δ (0.6∼0.8 is recommended for optimal performance in distinguishing knowledge boundaries) between orginal task defination P_{task} from P and LLMs-cognitive task defination P^*_{task} from P^*, and finally get a cognitive signal S_{cog}, formally:

$$S_{\text{cog}} = \begin{cases} unknown & , \texttt{sim}\left(\mathcal{M}_{\text{emb}}(P_{\text{task}}), \mathcal{M}_{\text{emb}}(P^*_{\text{task}})\right) < \delta \\ known & , \texttt{sim}\left(\mathcal{M}_{\text{emb}}(P_{\text{task}}), \mathcal{M}_{\text{emb}}(P^*_{\text{task}})\right) \geq \delta \end{cases} \tag{7}$$

where $\texttt{sim}(*)$ is a cosine similarity function that computes the similarity between two embedding vectors.

By efficiently evaluating cognitive results, *CPM* integrates alternative aggregation strategies for $\mathcal{M}_{\mathcal{LLM}}$ based on S_{cog} [1], as detailed in appendix A.3: ***(1) Solution logic aggregation for known tasks:*** $\mathcal{M}_{\mathcal{LLM}}$ merges beneficial aspects from the original prompt P with the LLM-taste prompt P^* to create the final prompt P_{final}. ***(2) Stylistic template aggregation for unknown tasks***:$\mathcal{M}_{\mathcal{LLM}}$ extracts a cognitive preference template for thinking T from the incorrect context in the LLM-taste prompt, and integrates meta-cognitive elements from the original prompt P into T to construct the final prompt P_{final}.

Consequently, we utilize the final LLM-preferred prompt to query the LLM $\mathcal{M}_{\mathcal{LLM}}$ with a specific problem input, $problem$, to obtain the final logical answer, $answer$:

$$answer = \mathcal{M}_{\mathcal{LLM}}(P_{final}, problem) \tag{8}$$

4 Experiments

4.1 Datasets and Tasks

Following Yao et al. (2024); Suzgun and Kalai (2024); Yang et al. (2024a), we evaluate RoT across eight tasks (detailed in Appendix A.1): **(1) Mathematical Reasoning**:Game of 24 (Yao et al., 2024; Xiang et al., 2025), Multilingual Grade School Math (Shi et al., 2022); **(2) Domain Knowledge**: Python Puzzles (Schuster et al., 2021), Geometric Shapes, Multi-Step Arithmetic Two, Word Sorting, and Checkmate-in-One (Srivastava et al., 2023); **(3) Creativity**: Shakespearean Sonnet Writing (Suzgun and Kalai, 2024).

4.2 Baselines

In our experiments, we compare RoT with five classic and latest state-of-the-art prompting baselines:

- **CoT Prompting:** Following Suzgun and Kalai (2024); Yang et al. (2024a), we employ `GPT-4` to decompose instruction into logic intermediate reasoning steps activated by *"Let's think step by step"*.
- **Meta-Prompting:** Suzgun and Kalai (2024) introduced general, task-agnostic prompts as a scaffold to guide LLMs effectively perform logic tasks.
- **Graph-of-Thought (GoT):** Besta et al. (2024) modeled reasoning as a graph, where nodes are thoughts and edges define their relationships to solve complex problems.
- **Tree-of-Thought (ToT):** Yao et al. (2024) organized reasoning in a tree structure, allowing LLMs to explore multiple thought paths and select the most promising ones for problem-solving, enhancing their complexity management in logical reasoning tasks.

[1] we use S_{cog} to conduct reverse evaluation of `GPT-4` in our experiments.

- **Buffer of Thought (BoT):** Yang et al. (2024a) introduced a meta-buffer that stores high-level thought templates, allowing for the adaptive retrieval and instantiation of relevant templates to enhance reasoning efficiency.

4.3 Experiment Setup

To ensure fair comparisons with previous methods, following Liu et al. (2024a); Yao et al. (2024); Zhang et al.; Suzgun and Kalai (2024), we utilize two LLMs: `GPT-3.5-turbo` and `GPT-4`, as the foundational model via OpenAI API for our RoT experiments including both main experiments and ablation study. For the $warmup$ hyperparameter, we experimented with values of $\{1, 3, 5, 10\}$, accessed in batches for reverse reasoning warm-up through OpenAI API. Our findings suggest that a value of 5 optimally balances between logical accuracy and cost-efficiency. For embedding model $\mathcal{M}_{emb}$, we utilize a huggingface model `stella_en_1.5B_v5` [2], a high-performance model with the smallest parameter countamong the top three on the `MTEB` leaderboard (Muennighoff et al., 2023), offline deployed on a single `NVIDIA GeForce RTX 4090 GPU`.

4.4 Evaluation Metrics

Knowledge Boundary and Cognitive Preference Consistency. As described in Section 3.3, we trigger knowledge boundary using S_{cog} by comparing $\texttt{sim}(P_{task}, \mathcal{M}_{\mathcal{LLM}}(D, P_r)_{task})$ with a threshold $\delta = 0.7$. We first evaluate `GPT-3.5-turbo` & `GPT-4` on experimental tasks to distinguish between known and unknown domains in both 1-shot and 2-shot settings[3] and subsequently conduct human evaluations of average cognitive preference consistency Con_{cog} based on stylistic and grammatical norms on P^* across various tasks, with three professional annotators validating cognitive preferences on a 1–5 scale.

Reasoning Accuracy. Following Suzgun and Kalai (2024); Yao et al. (2024), we introduce a LLM[4] to validate the final logical reasoning against gold results (Correct/Wrong). We then compute logical accuracy Acc_{logic} for each logical task by tallying the number of correct responses.

[2] https://huggingface.co/dunzhang/stella_en_1.5B_v5, License:mit

[3] To achieve better cost savings and fairness, we set the few-shot for all methods to 1-shot and 2-shot.

[4] we select the latest openai-o1 as judger, https://openai.com/index/introducing-openai-o1-preview/

Reasoning Efficiency. Following Yang et al. (2024a), we evaluate reasoning efficiency in terms of complexity by calculating T, calculated as the average time spent per task across all samples in the test dataset:

$$T = \frac{1}{N} \sum_{i=1}^{M} \sum_{j=1}^{N} T_{ij} \tag{9}$$

where N is the total number of tasks, M is the total number of samples, and T_{ij} represents the time taken for the i-th task on the j-th sample.

5 Results and Discussion

5.1 Knowledge Boundary and Cognitive Preference Consistency

Knowledge Boundary. As shown in Table 2, our findings provide preliminary support for our hypothesis concerning cognitive knowledge boundaries of LLMs in reverse reasoning across current task benchmarks in one-shot and two-shot settings (Yao et al., 2024; Suzgun et al., 2023; Srivastava et al., 2023; Schuster et al., 2021; Shi et al., 2022; Suzgun and Kalai, 2024). Notably, `GPT-3.5-turbo` & `GPT-4` excel in structured reasoning tasks such as `Game of 24`, `Geometric Shapes`, and `Checkmate-in-One`, consistently achieving strong results in both one-shot and two-shot settings. In contrast, tasks such as `MGSM (avg)` and `Python Puzzles` fall into an unknown category in the one-shot scenario, stemming from multi-source problems that are challenging to make reverse reasoning without sufficient context, indirectly supporting the rationale for our *CPM* approach.

Cognitive Preference Consistency. Our human evaluation of Con_{cog} yielded high scores of **4.32** in the 2-shot setting and **4.01** in the 1-shot setting, further validating that LLMs exhibit cognitive preferences shaped by RLHF (Ouyang et al., 2022).

5.2 Reasoning Accuracy and Efficiency

(1) RoT can activate LLMs' reasoning accuracy. As shown in Table 1, RoT consistently outperforms all baselines across various reasoning tasks, with particularly notable improvements observed in `GPT-4`. Specifically, compared with the best BoT, RoT achieves significant gains in tasks such as `Game of 24` **(+17.15% in 1-shot and +17.08% in 2-shot)**, `Geometric Shapes` **(+4.87% in 1-shot and +4.96% in 2-shot)**, and `Checkmate-in-One`

Baselines		Game of 24	Geometric Shapes	Multi-Step Arithmetic	Word Sorting	Checkmate-in-One	Python Puzzles	MGSM	Sonnet Writing
					GPT-3.5-turbo				
CoT (*NeurIPS*) Wei et al. (2022)	1-shot	26.3	50.2	73.9	71.9	25.7	30.1	75.0	58.6
	2-shot	29.6	53.2	76.2	74.6	25.4	30.6	77.4	61.3
Meta-prompting Suzgun and Kalai (2024)	1-shot	59.6	69.5	83.1	90.0	47.4	40.2	80.3	69.5
	2-shot	60.2	71.0	85.1	91.5	48.0	41.1	81.5	71.4
ToT (*NeurIPS*) Yao et al. (2024)	1-shot	66.7	56.2	81.9	89.2	44.2	38.2	78.3	63.5
	2-shot	67.3	58.5	82.3	90.3	45.2	39.5	79.4	65.3
GoT (*AAAI*) Besta et al. (2024)	1-shot	69.7	58.6	79.1	88.8	45.4	35.3	80.3	62.2
	2-shot	70.4	58.9	79.0	89.5	46.0	36.7	81.0	63.7
BoT (*NeurIPS*) Yang et al. (2024a)	1-shot	74.7	83.1	87.6	93.2	67.9	48.2	81.1	71.1
	2-shot	75.5	84.3	88.3	94.4	68.5	48.0	82.3	73.0
RoT (Ours)	1-shot	**82.8**	**88.4**	**89.2**	**95.2**	**71.5**	**50.2**	**82.3**	**75.5**
	2-shot	**87.8**	**88.7**	**89.5**	**95.6**	**72.6**	**50.4**	**84.7**	**75.8**
					GPT-4				
CoT (*NeurIPS*) Wei et al. (2022)	1-shot	31.3	57.4	83.1	80.7	35.8	35.3	83.9	66.3
	2-shot	32.7	60.9	85.5	82.7	36.7	35.9	84.3	70.2
Meta-prompting Suzgun and Kalai (2024)	1-shot	64.6	76.7	89.6	97.6	58.2	45.0	85.1	78.7
	2-shot	66.3	78.2	90.7	98.4	58.9	45.6	85.5	79.8
ToT (*NeurIPS*) Yao et al. (2024)	1-shot	73.7	58.6	88.8	95.2	49.8	43.8	84.7	68.7
	2-shot	74.5	60.1	90.3	96.0	48.8	45.2	86.3	69.8
GoT (*AAAI*) Besta et al. (2024)	1-shot	74.7	56.2	87.6	95.6	49.0	42.6	85.9	68.3
	2-shot	73.5	57.7	88.7	96.8	51.2	43.5	86.7	62.5
BoT (*NeurIPS*) Yang et al. (2024a)	1-shot	82.8	90.4	94.8	99.2	87.1	51.8	87.9	79.1
	2-shot	83.7	90.7	96.8	99.6	88.3	52.8	87.5	79.8
RoT (Ours)	1-shot	**97.0**	**94.8**	**98.4**	**99.6**	**91.2**	**54.6**	**88.7**	**89.2**
	2-shot	**98.0**	**95.2**	**99.2**	**100.0**	**92.0**	**57.3**	**90.0**	**92.0**

Table 1: Comparison of RoT with baselines across 1-shot and 2-shot settings for reasoning accuracy.

Task	1-shot		2-shot	
	Kno.	Unkno.	Kno.	Unkno.
Game of 24	✓		✓	
MGSM (avg)		✓	✓	
Multi-Step Arithmetic	✓		✓	
WordSorting	✓		✓	
Python Puzzles		✓	✓	
Geometric Shapes	✓		✓	
Checkmate-in-One	✓		✓	
Sonnet Writing	✓		✓	

Table 2: Knowledge Boundary under Reversal Reasoning for `GPT-3.5-turbo` and `GPT-4` with identical results. *Kno.* for Known, *Unkno.* for Unknown.

(+4.71% in 1-shot and +4.19% in 2-shot), demonstrating our flexibility of leveraging cognitive preference in LLM to activate logic capabilities through reverse reasoning (case study detailed in appendix B). For `GPT-3.5-turbo`, RoT also demonstrates substantial improvements, such as `Game of 24` **(+10.84% in 1-shot and +16.29% in 2-shot)** and `Geometric Shapes` **(+6.38% in 1-shot and +5.22% in 2-shot)**, further emphasizing its versatility in activating logic through cognitive preference management.

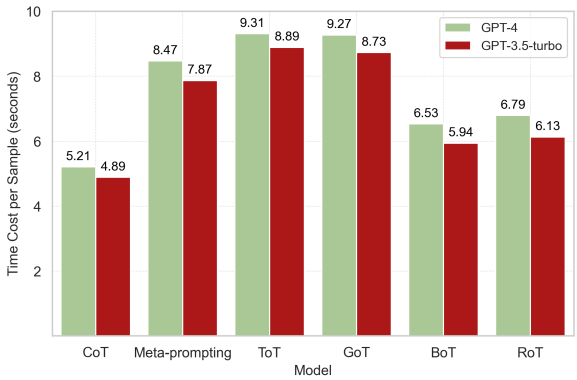

Figure 3: Inference time comparison

(2) RoT demonstrates better tradeoff between reasoning accuracy and efficiency across baselines. As shown in Figure 3, RoT achieves competitive performance in reasoning efficiency, outperforming baselines and being second only to BoT, while BoT's dependence on numerous pre-defined golden thought templates limits its flexibility. In contrast, our RoT, as a plugin strategy, emphasizes

Ablation study		Game of 24✓	Geometric Shapes✓	Multi-Step Arithmetic✓	Word Sorting✓	Checkmate-in-One✓	Python Puzzles×	MGSM×	Sonnet Writing✓
					GPT-3.5-turbo				
RoT	1-shot	**82.8**	**88.4**	**89.2**	**95.2**	**71.5**	**50.2**	**82.3**	**75.5**
	2-shot	**87.8**	**88.7**	**89.5**	**95.6**	**72.6**	**50.4**	**84.7**	**75.8**
w/o *PGRR*	1-shot	74.7	72.3	74.3	74.7	65.5	44.2	78.7	69.6
	2-shot	75.5	72.6	75.4	76.2	66.9	44.4	79.0	69.8
w/o *Logic*	1-shot	76.7	80.7	81.5	84.3	66.3	45.8	77.5	72.9
	2-shot	76.5	80.2	81.9	85.1	66.5	46.0	78.2	73.0
w/o *CPM*	1-shot	80.8	86.0	88.4	93.2	70.3	39.8	72.7	74.2
	2-shot	81.6	86.7	88.3	93.5	71.0	47.2	80.2	75.0
					GPT-4				
RoT	1-shot	**97.0**	**94.8**	**98.4**	**99.6**	**91.2**	**54.6**	**88.7**	**89.2**
	2-shot	**98.0**	**95.2**	**99.2**	**100.0**	**92.0**	**57.3**	**90.0**	**92.0**
w/o *PGRR*	1-shot	77.8	74.7	78.7	79.5	80.7	45.0	85.9	72.7
	2-shot	78.6	75.0	79.0	81.0	82.3	46.8	86.0	73.0
w/o *Logic*	1-shot	85.9	83.1	87.1	89.2	81.5	49.8	85.5	86.1
	2-shot	86.7	83.5	87.9	89.5	83.5	50.0	86.7	77.0
w/o *CPM*	1-shot	92.0	90.0	94.8	97.6	89.2	40.6	84.7	88.8
	2-shot	93.9	90.3	95.2	98.4	89.1	51.6	86.3	91.5

Table 2: Ablation study of RoT across various tasks in 1-shot and 2-shot settings. ✓ indicates known tasks in both 1-shot and 2-shot; × indicates unknown tasks in 1-shot.

exploring optimal prompt for solution and task data instantiation after reverse reasoning warm-up.

5.3 Ablation Study

As shown in Table 2, we conducted three ablation studies to evaluate key components: (1) **w/o *PGRR***: Removing Preference-Guided Reverse Reasoning(*PGRR*) for exploring LLMs-taste prompts; (2) **w/o *Logic***: Excluding mathematical logic for pseudo-code plan-and-solve; and (3) **w/o *CPM***: Eliminating Cognitive Preference Manage(*CPM*) for both known and unknown tasks.

Impact of *PGRR*. Excluding Preference-Guided Reverse Reasoning (*PGRR*) results in a significant reduction in overall task performance, as seen in tasks like the Game of 24 (98.0% to 78.6% for GPT-4, 87.8% to 75.5% for GPT-3.5-turbo) and WordSorting (100% to 81% for GPT-4, 95.6% to 76.2% for GPT-3.5-turbo), indicating **w/o *PGRR*** weaken model's cognition to task-specific requirements.

Impact of *Logic*. Removing mathematical logic from RoT leads to notable declines in tasks requiring structured problem-solving, such as Multi-Step Arithmetic (99.2% to 87.9% for GPT-4, 89.5% to 81.9% for GPT-3.5-turbo) and Checkmate-in-One (92.0% to 83.5% for GPT-4, 72.6% to 66.5% for GPT-3.5-turbo), underscoring **w/o *Logic*** negatively affects structured problem-solving for complex reasoning.

Impact of *CPM*. Lacking Cognitive Preference Manager (*CPM*) has a particularly pronounced effect on unknown tasks, with Python Puzzles dropping from 54.6% to 45.0% for GPT-4 and 50.2% to 39.8% for GPT-3.5-turbo, while MGSM decreased from 84.7% to 80.2% for GPT-3.5-turbo and from 90.0% to 86.3% for GPT-4, indicating **w/o *CPM*** weaken RoT's flexibility for tackling known tasks and unknown tasks [5].

6 Conclusion

In this paper, we propose Reversal of Thought (RoT), a novel and plug-and-play framework to enhance the logical reasoning capabilities of LLMs. By integrating reverse reasoning with meta-cognitive mechanisms and cognitive preference management, RoT improves reasoning accuracy and efficiency while minimizing computational costs, which leverages *Preference-Guided Reverse Reasoning* and *Cognitive Preference Manager*, which optimally aligns LLM reasoning processes with their cognitive preferences shaped by their pretraining and RLHF. Comprehensive experiments across diverse reasoning tasks demonstrate that RoT consistently outperforms state-of-the-art baselines in both known and unknown task scenarios, demonstrating the potential to expand knowledge boundaries through cognitive preference template. Our research provides valuable insights into future studies focused on further enhancing LLMs' reasoning capacities by dynamically exploring cognitive preferences for complex reasoning tasks.

[5] Please refer to Appendix B.1.1 for case study on known task and Appendix B.2.1 for case study on unknwon task.

Limitations

Reversal of Thought (RoT) introduce a reverse reasoning warm-up to activate cognitive preference for LLMs to enhance logic capabilities and introduce a cognitive preference manager to determine knowledge boundary and utilize cognitive preference for known and unknown tasks.

While RoT has performed exceptionally in logic accuracy and efficiency, We discuss major challenge in its reliance on two-shot demonstration inputs involving two distinct problem cases. We observed that RoT may struggles with one-shot learning in multi-source tasks. we partially and effectively mitigates this issue through the integration of *Cognitive Preference Manager (CPM)* and two-shot learning.

In future work, we aim to extend RoT's capabilities by incorporating In-Context Learning (ICL), which will allow for greater flexibility in adapting to varied contexts and improve its performance on more complex reasoning tasks. Furthermore, we believe that utilizing Reversal of Thought in teacher-student model distillation, which could further amplify the practical value of our approach.

Ethical Considerations

Since the datasets used in our experiments focus on pure mathematical and algorithmic reasoning, domain-specific knowledge, and literary creativity, which are all sourced from publicly available datasets (Yao et al., 2024; Suzgun et al., 2023; Srivastava et al., 2023; Schuster et al., 2021; Shi et al., 2022; Suzgun and Kalai, 2024; Xiang et al., 2025) and devoid of any personal privacy or sensitive ethical information. Therefore, we do not identify any immediate ethical concerns regarding our current work. Additionally, we conduct human evaluations of cognitive preference consistency with three anonymous professional annotators following our instructions (detailed in appendix C).

References

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

Konstantine Arkoudas. 2023. Gpt-4 can't reason. *arXiv preprint arXiv:2308.03762*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Guangsheng Bao, Hongbo Zhang, Linyi Yang, Cunxiang Wang, and Yue Zhang. 2024. Llms with chain-of-thought are non-causal reasoners. *arXiv preprint arXiv:2402.16048*.

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

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. 2023. Large language models as tool makers. *arXiv preprint arXiv:2305.17126*.

Qiguang Chen, Libo Qin, WANG Jiaqi, Jingxuan Zhou, and Wanxiang Che. Unlocking the capabilities of thought: A reasoning boundary framework to quantify and optimize chain-of-thought. 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, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

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

Damien S Fleur, Bert Bredeweg, and Wouter van den Bos. 2021. Metacognition: ideas and insights from neuro-and educational sciences. *npj Science of Learning*, 6(1):13.

Benjamin Franklin. 2005. Tell me and i forget. teach me and i remember. involve me and i learn.

Soochan Lee and Gunhee Kim. 2023. Recursion of thought: A divide-and-conquer approach to multi-context reasoning with language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 623–658.

Tongxuan Liu, Wenjiang Xu, Weizhe Huang, Xingyu Wang, Jiaxing Wang, Hailong Yang, and Jing Li. 2024a. Logic-of-thought: Injecting logic into contexts for full reasoning in large language models. *arXiv preprint arXiv:2409.17539*.

Yinhong Liu, Han Zhou, Zhijiang Guo, Ehsan Shareghi, Ivan Vulic, Anna Korhonen, and Nigel Collier. 2024b. Aligning with human judgement: The role of pairwise preference in large language model evaluators. *arXiv preprint arXiv:2403.16950*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2024. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36.

Katerina Margatina, Timo Schick, Nikolaos Aletras, and Jane Dwivedi-Yu. 2023. Active learning principles for in-context learning with large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5011–5034.

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023. Mteb: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037.

Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models.

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

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

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

Matthew Renze and Erhan Guven. 2024. Self-reflection in llm agents: Effects on problem-solving performance. *arXiv preprint arXiv:2405.06682*.

Tal Schuster, Ashwin Kalyan, Alex Polozov, and Adam Tauman Kalai. 2021. Programming puzzles. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, et al. 2022. Language models are multilingual chain-of-thought reasoners. In *The Eleventh International Conference on Learning Representations*.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

Kaya Stechly, Matthew Marquez, and Subbarao Kambhampati. Gpt-4 doesn't know it's wrong: An analysis of iterative prompting for reasoning problems. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*.

Hongda Sun, Weikai Xu, Wei Liu, Jian Luan, Bin Wang, Shuo Shang, Ji-Rong Wen, and Rui Yan. 2024. Determlr: Augmenting llm-based logical reasoning from indeterminacy to determinacy. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9828–9862.

Mirac Suzgun and Adam Tauman Kalai. 2024. Meta-prompting: Enhancing language models with task-agnostic scaffolding. *arXiv preprint arXiv:2401.12954*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, et al. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051.

Yuan Tian, Nan Xu, and Wenji Mao. 2024. A theory guided scaffolding instruction framework for llm-enabled metaphor reasoning. 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 7731–7748.

Lucina Q Uddin. 2021. Cognitive and behavioural flexibility: neural mechanisms and clinical considerations. *Nature Reviews Neuroscience*, 22(3):167–179.

Xinglin Wang, Shaoxiong Feng, Yiwei Li, Peiwen Yuan, Yueqi Zhang, Boyuan Pan, Heda Wang, Yao Hu, and Kan Li. 2024. Make every penny count: Difficulty-adaptive self-consistency for cost-efficient reasoning. *arXiv preprint arXiv:2408.13457*.

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

Violet Xiang, Charlie Snell, Kanishk Gandhi, Alon Albalak, Anikait Singh, Chase Blagden, Duy Phung, Rafael Rafailov, Nathan Lile, Dakota Mahan, et al. 2025. Towards system 2 reasoning in llms: Learning how to think with meta chain-of-though. *arXiv preprint arXiv:2501.04682*.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E Gonzalez, and Bin Cui. 2024a. Buffer of thoughts: Thought-augmented reasoning with large language models. *arXiv preprint arXiv:2406.04271*.

Xiaocheng Yang, Bingsen Chen, and Yik-Cheung Tam. 2024b. Arithmetic reasoning with llm: Prolog generation & permutation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 699–710.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Xunjian Yin, Xu Zhang, Jie Ruan, and Xiaojun Wan. 2024. Benchmarking knowledge boundary for large language model: A different perspective on model evaluation. *arXiv preprint arXiv:2402.11493*.

Xiao Yu, Baolin Peng, Michel Galley, Jianfeng Gao, and Zhou Yu. 2024. Teaching language models to self-improve through interactive demonstrations. 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 5127–5149.

Yifan Zhang, Jingqin Yang, Yang Yuan, and Andrew C Yao. Cumulative reasoning with large language models.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, et al. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

Siyuan Zhou, Xinran Xu, Xiangyu He, Faxin Zhou, Yu Zhai, Jinglu Chen, Yuhang Long, Lifen Zheng, and Chunming Lu. 2023. Biasing the neurocognitive processing of videos with the presence of a real cultural other. *Cerebral Cortex*, 33(4):1090–1103.

A Experimental Details

A.1 Datasets and Tasks

To comprehensively validate our Reverse of Thought (RoT), following ToT (Yao et al., 2024), meta-prompting (Suzgun and Kalai, 2024) and BoT (Yang et al., 2024a), we assess it with baselines across a broad range of eight tasks across five logical benchmarks that encompass mathematical and algorithmic reasoning, domain-specific knowledge, and literary creativity:**(1) `Game of 24`** (Yao et al., 2024; Xiang et al., 2025) challenges LLMs to create a mathematical expression utilizing each of four given numbers exactly once to achieve 24[6]. **(2) `BIG-Bench`** (Suzgun et al., 2023; Srivastava et al., 2023) involves **Geometric Shapes**, **Multi-Step Arithmetic Two** and **Word Sorting** and **Checkmate-in-One** from BIG-Bench suite [7] (Srivastava et al., 2023); **(3) `Python Puzzles`** (Schuster et al., 2021) comprises a collection of challenging programming puzzles crafted in Python, covering various difficulty levels[8];**(4) `Multilingual Grade School` Math (MGSM)** (Shi et al., 2022) is a multilingual adaptation of the `GSM8K` dataset (Cobbe et al., 2021), featuring translations of a subset of examples into ten diverse languages[9] .**(5) `Shakespearean Sonnet Writing`** (Suzgun and Kalai, 2024) require LLMs to compose with the rhyme scheme `"ABAB CDCD EFEF GG"` while incorporating three [10] or five [11] specified words verbatim.

[6] https://www.4nums.com/game/

[7] https://huggingface.co/datasets/google/bigbench

[8] https://github.com/microsoft/PythonProgrammingPuzzles

[9] https://github.com/google-research/url-nlp/tree/main/mgsm

[10] https://huggingface.co/datasets/turingmachine/meta-prompting

[11] https://github.com/iljones00/Shakespearean-Sonnets-GPT

A.2 Prompt for Reverse Reasoning

Prompt for Reverse Reasoning

Instruction
You are a highly distinguished expert in mathematics and information reasoning. Based on the given example, define the specific task, including the task definition, pseudocode, logical pseudocode, case examples, and input-output format.
1. Understand Task Description:
Meticulously study demonstrations to deeply understand generic task description.
2. Plan Generic Pseudocode:
Provide pseudocode in text form and plan an efficient algorithm to complete the task with your experiences.
3. Formulate Logical Pseudocode:
Convert the pseudocode into generic logical algorithm pseudocode using ONLY logical symbols:
Logical Operators*:*
Conjunction: $A \wedge B$ *; Disjunction:* $A \vee B$
equivalence: $A \equiv B$ *, Negation:* $\neg A$
Quantifiers:
Universal quantifier: $\forall x$*; Existential quantifier:* $\exists x$
Inequalities:
Less than: $x < y$*; Greater than:* $x > y$
Less than or equal to: $x \leq y$
Greater than or equal to: $x \geq y$
Equals: $x = y$*; Not equals:* $x \neq y$
Conditional Statements:
If A *then* B*:* $A \supset B$
If $A \wedge B$ *then* C*:* $(A \wedge B) \supset C$
If $A \vee B$ *then* C*:* $(A \vee B) \supset C$
If $\forall x(P(x))$ *then* Q*:* $\forall x(P(x)) \supset Q$
If $\exists x(P(x))$ *then* Q*:* $\exists x(P(x)) \supset Q$ *etc.*
Input: *[Demonstration]* ***Output:*** *[Output]*

Figure 4: Prompt for Reverse Reasoning

A.3 Prompt for CPM (Known/Unknown)

Prompt for CPM (Known/Unknown)

Instruction
You are an expert in information synthesis, proficient in combining complementary insights and extracting essential details from the viewpoints of the distilled task definition, detailed generic logical pseudocode, case example, and input-output format.
1. For Known:
The Reversal Prompt and Benchmark Prompt should complement each other. Analyze the content and structure of both prompts to identify their accuracy, similarities and differences. Synthesize the key points and integrate them into a unified and coherent output.
2. For Unknown:
Extract a cognitive preference template T from any inaccuracies in the LLM-taste prompt. Integrate meta-cognitive elements from the original prompt P into this template to enhance T.
Input:
LLM-Taste Prompt:[LLM-Taste Prompt]
Benchmark Prompt:[Benchmark Prompt]
Output: *[Output]*

Figure 5: Prompt for CPM (Known/Unknown)

B Case Study

B.1 Case Study for Known Task (Game of 24)

B.1.1 P_{final} Case study for `Game of 24`

P_{final} Case study for known task after CPM

Task Defination

The task is to find a feasible mathematical expression using the four input numbers (a, b, c, d) in the order they are given, such that the result equals 24. The feasible solution should involve basic arithmetic operations (+, -, ×, /) and parentheses if necessary.

Logical Pseudocode

For each permutation $P(A, B, C, D)$ in permutations of [A, B, C, D]:

For each operation $op_n \in (+, -, \times, \div)$, where $n \in \{0, 1, 2, 3\}$:

- *For each permute $\exists(Expression == 24) \vee (Expression = permute[0]\ op_1\ permute[1]\ op_2\ permute[2]\ op_3\ permute[3])$:*
- *If $\exists$(Expression $== 24$), then:*
  - *Print(Expression)*
  - *Terminate the process.*
- *Else, continue checking the next permutation.*

If no solution is found, print "No feasible solution exists.".

Case Examples:

Example 1: *Input: 4 6 7 1 Output: One feasible solution (e.g., $6/(1-(4/7)) = 24$).*

Example 2: *Input: 4 2 2 1 Output: "No feasible solution exists."*

Input-Output Format:

Input: *Four integers separated by space (e.g., "3 3 8 8").*

Output: *If a feasible solution exists, output a string indicating one possible solution (e.g., "One feasible solution (e.g., $8 \div (3 - 8 \div 3) = 24$)."). If no solution exists, output the string "No feasible solution exists."*

B.1.2 Instantiation Case Study for `Game of 24` Using P_{final}

Instantiation Case Study for `Game of 24` Using P_{final}

Input: *4 9 10 13*

Steps: *Let me compute possible expressions for this input that result in 24 using provided logic pseudocode.*
1. Generate all permutations of the numbers (4,9,10,13): This gives 24 possible arrangements.
*2. Apply operations: For each permutation, test all combinations of the operations (+, -, *, /) between the numbers.*
3. Check if the result equals 24: Evaluate the expression and check if it equals 24.
4. Output the result: If a valid expression is found, print it. If not, output that no feasible solution exists.

```
import itertools
inputs = [4,9,10,13]
operations = ['+', '-', '*', '/']
def evaluate_expression(expr):
    try:
        return eval(expr)
    except ZeroDivisionError:
        return None
permutations = itertools.permutations(inputs)
op_combinations = itertools.product(operations, repeat=3)
solutions = []
for perm in permutations:
    for ops in op_combinations:
        exprs = [
        f"({perm[0]} {ops[0]} {perm[1]}) {ops[1]} ({perm[2]} {ops[2]} {perm[3]})",
        f"({perm[0]} {ops[0]} ({perm[1]} {ops[1]} {perm[2]})) {ops[2]} {perm[3]}",
        f"{perm[0]} {ops[0]} ({perm[1]} {ops[1]} ({perm[2]} {ops[2]} {perm[3]}))",
        f"({perm[0]} {ops[0]} {perm[1]}) {ops[1]} ({perm[2]} {ops[2]} {perm[3]})",
            ...
        ]
        for expr in exprs:
            if evaluate_expression(expr) == 24:
                solutions.append(expr)
print(solutions)
```

Output: 4 * (9 + (10 - 13))=24

B.2 Case Study for Unknown Task (MGSM)

B.2.1 P_{final} Case study for MGSM

P_{final} Case study for unknown task after CPM

Task Defination

The task is to solve a math problem text with a description of a situation. Your task is to calculate the answer and provide it in the format specified.

Logical Pseudocode

Input u from the user in natural language.

Parse u for numbers and contextual information.

Extract numbers α, β, γ, etc.

Identify contextual clues and operations, such as addition (+), subtraction (-), multiplication (*), division (/), and other implicit operations (e.g., percentages, halves, totals).

For each identified operation:

If operation is related to percentages, interpret $\times$ or division as needed.

If operation is additive (+), perform $Add(\alpha, \beta)$.

If operation is subtractive (-), perform $Subtract(\alpha, \beta)$.

If operation is multiplicative (*), perform $Multiply(\alpha, \beta)$.

If operation is divisive (/), perform $Divide(\alpha, \beta)$.

If implicit operations (e.g., "half that much"), interpret accordingly.

Handle complex structures, such as total amounts or remainders:

Use context to evaluate remaining quantities (e.g., after consumption, sales).

Apply operations in sequential order based on context.

Calculate the final result based on all interpreted operations and numbers.

Return the final result as a natural number or in currency (if applicable).

Case Examples:

Example 1:

Input: *A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take?*

Output: *3*

Input-Output Format:

Input: *A natural language string describing a scenario that involves certain arithmetic operations.*

Output: *A single integer or float number that is the result of the operations described in the scenario.*

B.2.2 Instantiation Case Study for MGSM Using P_{final}

Instantiation Case Study for MGSM Using P_{final}

Input: *Paul is at a train station and is waiting for his train. He isn't sure how long he needs to wait, but he knows that the fourth train scheduled to arrive at the station is the one he needs to get on. The first train is scheduled to arrive in 10 minutes, and this train will stay in the station for 20 minutes. The second train is to arrive half an hour after the first train leaves the station, and this second train will stay in the station for a quarter of the amount of time that the first train stayed in the station. The third train is to arrive an hour after the second train leaves the station, and this third train is to leave the station immediately after it arrives. The fourth train will arrive 20 minutes after the third train leaves, and this is the train Paul will board. In total, how long, in minutes, will Paul wait for his train?*

Steps:

Step 1: Symbol Information

1. α (first train arrival) = 10 minutes
2. β (first train stay) = 20 minutes
3. γ (second train stay as a quarter of the first train's stay) = $\frac{\beta}{4}$
4. δ (time between second train departure and third train arrival) = 60 minutes
5. ε (fourth train arrival after third train leaves) = 20 minutes

Step 2: Identify Contextual Clues and Operations

1. First train departure time = $\alpha + \beta$
2. Second train arrival time = First train departure time + 30 minutes
3. Second train stay time = γ
4. Third train arrival time = Second train departure time + δ
5. Fourth train arrival time = Third train departure time + ε

Step 3: Perform Calculations Based on Identified Operations

1. First train departure time = Add(α, β) $= 10 + 20 = 30$ minutes
2. Second train arrival time = Add(First train departure time,30) = $30 + 30 = 60$ minutes
3. Second train stay time = Divide($\beta, 4$) $= \frac{20}{4} = 5$ minutes
4. Second train departure time = Add(Second train arrival time ,Second train stay time) = $60 + 5 = 65$ minutes
5. Third train arrival time = Add(Second train departure time,δ) = $65 + 60 = 125$ minutes
6. Third train departure time = Third train arrival time (since it leaves immediately) = 125 minutes
7. Fourth train arrival time = Add(Third train departure time, ε) = $125 + 20 = 145$ minutes

Step 4: Calculate the Final Result

Final Result: Fourth train arrival time = 145 minutes

Output: 145 minutes

C Instructions for Human Evaluation.

To evaluate the cognitive preference consistency Con_{cog} between LLMs-taste prompt for task cognitions across different tasks, annotators adhere to the following guidelines:

Criteria. **(1) Style Consistency**: How well responses maintain a consistent stylistic tone for thinking across different tasks. **(2) Coherence**: Internal logic and smooth flow of the response.

Scoring. Responses are evaluated on a scale from 0 to 5. A score of **4-5** signifies exemplary style consistency, while **3-4** indicates strong consistency with minor variations. Scores of **2-3** represent adequate style but with noticeable inconsistencies, and **1-2** denote inconsistent style across tasks. A score of **0-1** reflects complete inconsistency.