Chain of Methodologies: Scaling Test Time Computation without Training

Anonymous ACL submission

Abstract

Large Language Models (LLMs) often struggle with complex reasoning tasks due to insufficient in-depth insights in their training data, which are typically absent in publicly available documents. This paper introduces the Chain of Methodologies (CoM), an innovative and intuitive prompting framework that enhances structured thinking by integrating human methodological insights, enabling LLMs to tackle complex tasks with extended reasoning. CoM leverages the metacognitive abilities of advanced LLMs, activating systematic reasoning throught user-defined methodologies without explicit fine-tuning. Experiments show that CoM surpasses competitive baselines, demonstrating the potential of training-free prompting methods as robust solutions for complex reasoning tasks and bridging the gap toward human-level reasoning through human-like methodological insights.

1 Introduction

Recently, OpenAI's o1 (OpenAI, 2024) showcases the possibility of using a long chain of thoughts to improve the reasoning ability of Large Language Models (LLMs). During these long thoughts, OpenAI's o1 displays high-level cognitive abilities, such as problem decomposition, error recognition, and correction, which constantly steer the thoughts in the right direction. OpenAI confers o1 with such abilities through reinforcement learning.

This paper explores whether LLMs can achieve similar self-guiding abilities for long, structured reasoning across domains using only prompts, without instruction fine-tuning. This is a challenging problem: while fine-tuning with large datasets can broadly improve instruction-following, conventional prompts are typically limited to specific tasks with few-shot examples due to constraints like context length and information extraction accuracy. As a result, pure prompting methods are rarely used for cross-domain tasks, despite their advantages—low cost, rapid deployment, high sample efficiency, and avoidance of catastrophic forgetting or data bias.

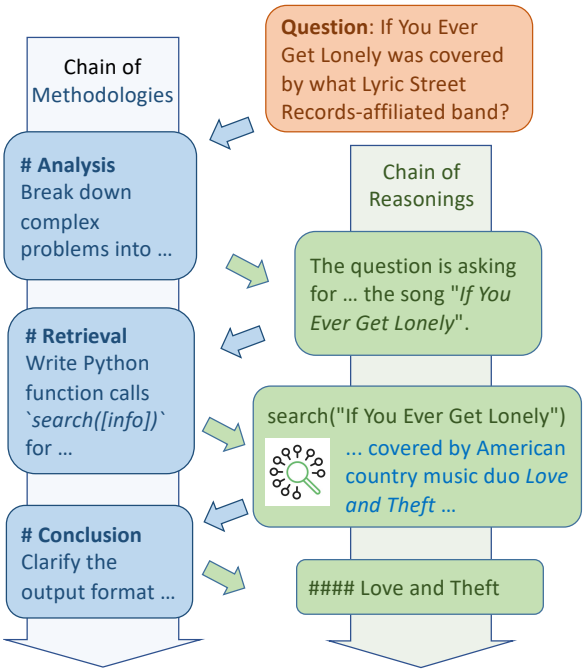

Figure 1: An Example of our Chain of Methodologies reasoning process, where the generation of methodologies and reasoning interleaves. A methodology (in blue) is selected based on the historical reasoning status, while the next reasoning step (in green) is guided by the previously selected methodology.

Our work is inspired by prior research on metacognitive knowledge in LLMs, which refers to the ability to reason about one's own reasoning processes. Pedagogical studies show that enhancing metacognitive knowledge improves reasoning in humans, and similar benefits have been observed in LLMs through prompts encouraging introspection and self-reflection (Wang and Zhao, 2024). Besides, Microsoft's Phi-3 (et al, 2024) uses system prompts like "do not hallucinate" to reduce hallucination, while (Didolkar et al., 2024) shows im-

proved mathematical reasoning when LLMs identify required skills to retrieve relevant examples. These findings provide both intuitive and empirical support for our approach.

We propose the Chain of Methodologies (CoM), an intuitive task-agnostic prompting technique designed to enable cross-domain self-guided reasoning without instruction fine-tuning. CoM uses methodology as catalysts to stimulate LLMs to generate the next reasoning step based on the reasoning history. While LLMs often struggle with complex reasoning tasks due to insufficient in-depth insights between problems and their respective solutions in the training data, CoM bridges this gap and enables smooth transitions from a problem to its solution by inserting a methodological analysis before each solution step. This leverages the metacognitive knowledge of LLMs to select or generate methodologies that justify or explain the next steps.

CoM features two key components: (1) a list of methodologies formatted in our "when-what" format, which facilitates selection based on the reasoning history and connects it to the next reasoning step, and (2) a methodology-reasoning loop that iteratively selects the next methodology to guide reasoning along an extended and well-structured reasoning path. An example of CoM's interleaving methodology-selecting and reasoning path is illustrated in Figure 1. Two examples of user-defined methodologies are listed in Figure 3.

Our contributions include the simple CoM framework and extensive experiments. CoM produces structured, explainable, and faithful reasoning paths. It is also highly extensible in that users can enhance the framework by modifying the list of methodologies in plain text. We evaluated our task-agnostic CoM framework on two types of representative and challenging tasks: mathematical reasoning and retrieval-augmented generation. Experiments show that CoM outperforms competitive baselines on these tasks across diverse LLMs.

2 Chain of Methodologies

2.1 Overview

We aim to use prompts to stimulate high-level cognitive (metacognitive) knowledge in existing LLMs, enabling them to possess the same cross-domain self-guiding ability as OpenAI's o1 thereby successfully carrying out extended and structured reasoning sequences across various domains. These prompts should be task-agnostic and effective in

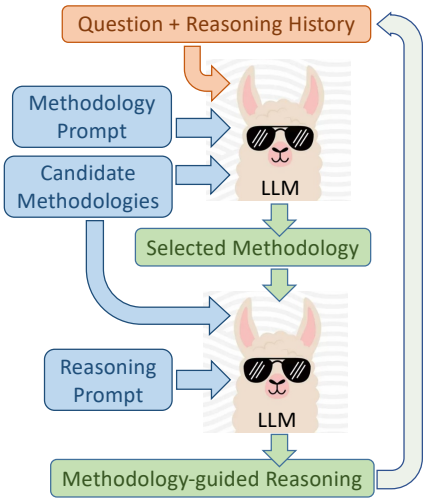

Figure 2: The components in our CoM framework and their interactions in the methodology-reasoning iterations.

guiding thought processes. We find prompts related to methodology are ideal candidates for this purpose. Methodology is a critical component of any discipline or field that requires a structured approach to understanding, problem-solving, or conducting research. It provides a framework that ensures tasks are executed consistently and effectively.

Our Chain of Methodologies (CoM) framework consists of a list of user-defined methodologies and a methodology-reasoning iteration. Each methodology provides a guideline for the next reasoning step based on the reasoning history. The reasoning process of CoM alternates between a methodology selection step and a methodology-guided reasoning step, as illustrated in Figure 1.

List of Methodologies: Unlike AlphaGo, which operates within a defined set of rules and a closed action space, the general problem-solving ability of human in an open action space is more complex and challenging to optimize. In fact, the accumulation and evolution of human methodologies have relied on fundamental processes such as trial and error, reflection, and self-correction based on problem-solving experiences across different eras and civilizations. To navigate this complexity, we integrate human knowledge and experience related to task completion through established methodologies. Let $M = \{m^{(1)}, m^{(2)}, \cdots, m^{(n)}\}$ denote the list of n user-defined methodologies.

Reasoning iterations: CoM conduct a maximum of K steps for each question Q. In step k, where $1 \leq k \leq K$, we first prompt an LLM_m

Analysis
- When: In step 1.
- What: Analyze the category and solution type of the question, list the key facts, variables, relations, constraints with their associated values, and clarify the required output format. Break down complex problems into simpler steps while maintaining critical context. Propose a sequence of methodologies necessary to tackle the remaining reasoning steps iteratively and explain how they are related to the final result.

Retrieval
- When: Fact-based information from the internet is needed.
- What: Write 1-3 line(s) of Python function call(s) `search([information],topk=3)` for each information needed to retrieve. The function `search` has been defined and imported for you, which returns a text summary for the argument `information`. Place your code in a single ```python\n...``` code-block. Finally, accurately simulated the retrieved output by yourself.

Figure 3: Two example methodology definitions in our *when-what* format.

with prompt template P_p to select a methodology $m_k \in M$ based on the reasoning history h_k:

$$m_k = LLM_p(M, Q, h_k, P_p) \quad (1)$$

, and then prompt an LLM_r with prompt template P_r to generate the next reasoning sequence r_k based on methodology m_k and history h_k:

$$r_k = LLM_r(M, Q, h_k, m_k, P_r) \quad (2)$$

, where the reasoning history contains all previous reasoning sequences $h_k = [r_1, r_2, \cdots, r_{k-1}]$. In this paper, we simply use the same instruction fine-tuned LLM for both LLM_m and LLM_r, which is frozen during the application of our framework.

2.2 Methodology Definition

Our emphasis is on a framework that utilizes a user-defined list of methodologies rather than studying the philosophy of finding a universally applicable set of methodologies, whose existence is a debated topic between universalism and contextualism. From a pragmatic standpoint, we focus on representing each methodology in a way that facilitates methodology selection and methodology-based reasoning.

To clarify the distinction between method and methodology, a method refers a specific technique or systematic procedure for accomplishing a task, whereas a methodology encompasses the principles and rationale guiding the selection and application of methods. Each methodology in our user-defined list specifies two key fields: *when* and *what*. The *when* field defines the applicable stage of the methodology in the reasoning lifecycle, along with the context and factors influencing the choice of the methodology. The *what* field outlines the systematic approach, action selection criteria, and expected outcomes of the methodology.

Specifically, a methodology is defined in Markdown format with three fields: (1) its name, (2) *when*: the situation and timing for its application, and (3) *what*: its specification and details, including principles, tools, techniques, and procedures. Figure 3 provides examples of two methodology definitions.

Next, we discuss different types of methodologies. We categorize methodology definitions into three broad types: *analysis*, *coding*, and *reflection*. *Analysis* methodologies guide the LLM in organizing information, such as extracting facts, variables, relations, constraints, and objectives from the question; breaking down the initial question into manageable sub-problems; planning the sequence of actions; and summarizing, rearranging, and distilling the information obtained so far. *Coding* methodologies prompt the LLM to generate formal languages for execution by solvers to obtain accurate results, or to use external tools (e.g., search engines) by calling predefined functions attached to the solvers. *Reflection* methodologies encourage the LLM to identify errors and provide constructive feedback through self-reflection or self-verification, enabling adjustments to the approach and proposing alternative strategies for subsequent steps. Figure 8 in Appendix A.1 lists the task-agnostic methodology definitions we used in our experiments.

In summary, the use of methodologies serves a multifaceted purpose: (1) providing human-input methodologies to stimulate the metacognitive ability of LLMs, compensating for the lack of in-depth insights in their training data for complex reasoning

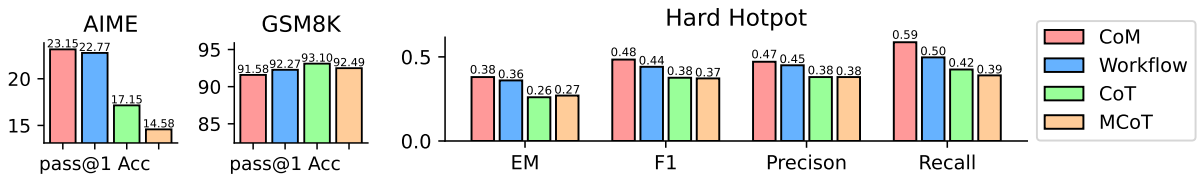

Figure 4: Results of Qwen2-72B-Instruct on AIME, GSM8K, and Hard HotpotQA.

task; (2) establishing a natural connection through explanation or justification between the current reasoning situation and its solution in the next step; and (3) offering an educated guess for the next step, avoiding the formidable complexity of stochastic search methods like MCTS (Qi et al., 2024) and RL (OpenAI, 2024; Snell et al., 2024; Zelikman et al., 2022), which operate over a general reasoning space that is much larger than those in games like AlphaGo.

Finally, our framework is designed for easy extensibility: users can update the list of methodology definitions in plain text to make it more comprehensive for general thinking or tailor it to a specific set of skills that accurately target a particular task.

2.3 Methodology-Reasoning Iterations

As illustrated in Figure 2, CoM alternates between prompting the LLM to generate the next methodology and the next methodology-based reasoning sequence for a maximum of K iterations.

The first prompt instructs the LLM to select a methodology for the next reasoning steps. This prompt concatenates the list of user-defined methodology definitions, the question, the history of previous methodology-based reasoning sequences, and an instruction that provides additional information about the reasoning stage and the output format. It enables the LLM to choose the most suitable methodology for the task.

The second prompt includes all the information from the first prompt, along with the methodology selected using the first prompt. It directs the LLM to adhere the guidance outlined in the chosen methodology while reasoning. Additionally, the second prompt requires the output to include the following elements: (1) an acknowledgment of the selected methodology by restating its name, (2) a chain-of-thought reasoning process or a program that implements the methodology, and (3) a summarized result of the reasoning or a guessed output of the program.

Following the second prompt, a solver is invoked to post-process the LLM's output. This step facilitates the LLM's programming ability (Chen et al., 2023). Currently, we have only implemented a Python interpreter, which is triggered when Python code blocks are detected in the output. This interpreter executes the code in a secure environment with several common packages pre-imported. After execution, the predicted output of the program in the LLM's response is replaced with the actual stdout output from the code's execution. This approach ensures accurate reasoning on tasks that require computation, such as mathematical tasks, effectively implementing the human methodology: "You should use a calculator for tasks that involve complex calculations." Furthermore, it enables various types of tool-using via Python APIs during the reasoning process, including web searches, knowledge base retrieval, and even invocation of other LLMs or manipulation of the LLM's own reasoning process (Cao et al., 2023).

Our Python interpreter executes code in a sandboxed environment, which operates as a new process with a safe global scope. In this environment, the code can only access a limited set of built-in functions and import from a predefined list of packages. We enforce a timeout of 1 minute for each process, as we empirically determined that larger timeouts do not significantly improve performance on our experimental tasks. Users can extend the tool-using capabilities of the CoM framework by adding corresponding methodology definitions and implementing relevant functions in the Python interpreter. For instance, to enable Google search, one could add a methodology definition specifying the existence of a function named "search" and the meaning of its arguments, followed by implementing and adding this function to the global scope of the Python interpreter.

Our prompts for methodology selection and methodology-based reasoning are provided in Figure 7 in the Appendix A.1.

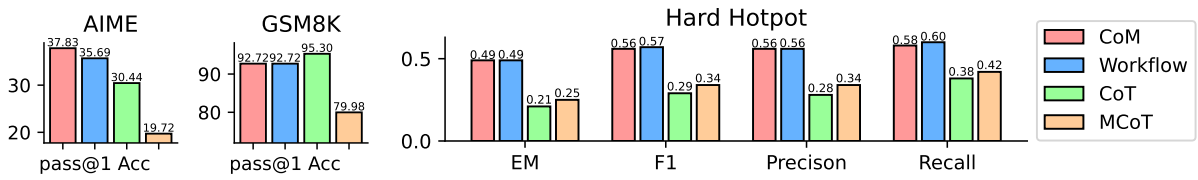

Figure 5: Results of DeepSeek-V3 on AIME, GSM8K, and Hard HotpotQA.

3 Related Work

Prompting A significant body of work has explored various prompt designs to enhance the reasoning capabilities of LLMs. Notable approaches include Chain-of-Thought (Wei et al., 2022), Least-to-Most (Zhou et al., 2023), Self-Consistency (Wang et al., 2023b), and Tree-of-Thoughts (Cao et al., 2023). Methods to enhance problem-specific performance, include question rephrasing, dividing subtasks, verification, symbolic grounding (Lyu et al., 2023; Xu et al., 2024a; Wang et al., 2023a; Zelikman et al., 2022; Wang et al., 2024), factuality and faithfulness verification for reasoning chains (Wang et al., 2024), as well as explicit separation of knowledge retrieval and reasoning steps to organize decision-making (Jin et al., 2024).

Iterative Prompting Prior research has also investigated iterative prompting methods to structure reasoning processes. Examples include Self-Refine (Madaan et al., 2023), IRCoT (Trivedi et al., 2023), iCAP (Wang et al., 2022), MetaGPT (Hong et al., 2024), and Chain of Ideas (Anonymous, 2024b). These approaches typically rely on predefined, hardcoded actions to guide reasoning. In contrast, our work introduces a task-agnostic framework that leverages the metacognitive abilities of LLMs to dynamically select methodologies based on reasoning history. Furthermore, while prior work focuses on generating the next reasoning step, our approach adopts a justification-before-action style, where the model introspectively justifies why a specific methodology is needed before executing it. This mirrors human metacognitive processes and distinguishes our work from implicit context-aware token generation.

Metacognition-based Several contemporary works are closely related to our approach. Buffer of Thoughts (Yang et al., 2024c) derives high-level guidelines from previously completed tasks and stores them in a buffer for future reuse, enabling learning from experience and improving efficiency by distilling level-2 slow thinking into level-1 fast thinking. However, unlike our work, its high-level guidelines contain problem-specific reasoning chains or code templates tailored to particular tasks, such as complex multi-query tasks. Skill-based CoT (Didolkar et al., 2024) explores the metacognitive capabilities of LLMs in mathematical problem-solving by labeling questions with corresponding skills, clustering them to reduce redundancy, and retrieving skill-relevant examples for in-context learning during inference. Induction-augmented generation (Zhang et al., 2023b) identifies key concepts in questions and uses inductive prompting templates to extract their close concepts and common attributes, facilitating more accurate reasoning processes.

Search-based rStar (Qi et al., 2024) introduces a self-play mutual reasoning approach that significantly improves the reasoning capabilities of small language models without fine-tuning. This method employs a costly Monte Carlo Tree Search (MCTS) with a set of five reasoning-inducing prompts.

Training-based Finally, training-based methods have been developed to enable LLMs to handle long chains of thought. For example, STaR (Zelikman et al., 2022) demonstrates that iterative training on reasoning histories leading to correct answers enables models to solve increasingly complex problems. Similarly, (Snell et al., 2024) fine-tunes small models to perform more reasoning steps using reinforcement learning with beam search, lookahead search, and best-of-N verifiers. ReST-MCTS (Zhang et al., 2024) integrates process reward guidance with tree search MCTS to collect higher-quality reasoning traces, while AFlow (Anonymous, 2024a) iteratively refines task-specific workflows. These methods highlight the potential of training-based approaches but often require significant computational resources.

4 Experiments

4.1 Experiment Setup

We evaluate the effectiveness of two components in CoM: methodology selection and methodology-guided reasoning.

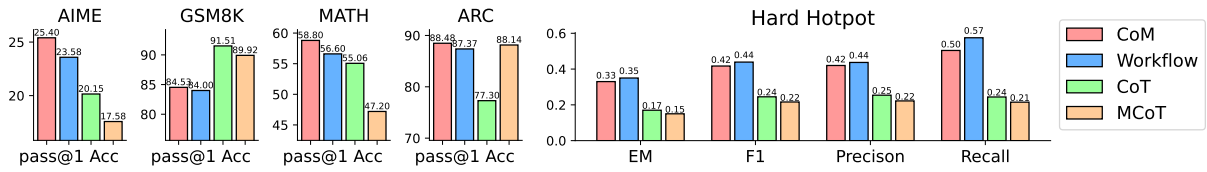

Figure 6: Results of Qwen2.5-7B-Instruct, a small LLM, on AIME, GSM8K, MATH, ARC, and Hard HotpotQA.

Table 1: LLMs used in our experiments. Results on the last three LLMs are reported in Appendix A.2.

LLM	Size
DeepSeek-V3 (DeepSeek-AI, 2024)	671B
Qwen2-72B-Instruct (Yang et al., 2024a)	72B
Qwen2.5-7B-Instruct (Yang et al., 2024a)	7B
Macro-o1 (Zhao et al., 2024)	7B
Yi-1.5-9B-Chat (01.AI: Alex Young, 2024)	9B
InternLM2.5-7B-chat (Cai et al., 2024)	7B
GLM-4-9b-chat (GLM, 2024)	9B

LLMs As listed in Table 1, we report experiment results conducted on a relatively large and small LLMs as well as a recent open-source model reminiscent of OpenAI's o1, named Macro-o1, which is a fine-tuned Qwen2-7B-Instruct with a combination of the filtered Open-O1 CoT dataset (Team, 2024), Macro-o1 CoT dataset, and Macro-o1 Instruction dataset. We use the LLM API provided by Siliconflow (sil) and Baidu Cloud (clo), with settings: max_tokens=1024, temperature=0.2, top_k=40, top_p=0.95, n=1.

Dataset We evaluate CoM using the same methodology definitions (Figure 8) on the test splits of the datasets listed in Table 2.

AIME: The 1983-2024 part of the American Invitational Mathematics Examination, includes complex algebraic equations, geometric puzzles, and advanced number theory problems to assess mathematical understanding and problem-solving skills.

GSM8K: Linguistically diverse grade school math word problems requiring 2 to 8 steps of elementary calculations to solve.

MATH-500: 500 problems from the MATH benchmark created by OpenAI.

HotpotQA: The hard portion of a multi-hop, multi-step QA dataset. We simulate retrieval-augmented generation (Anonymous, 2025) by presenting only the question to LLMs. When LLMs generate code calling a search with keywords, we use fuzzy string matching to retrieve the top-k most similar supporting facts.

ARC: AI2's Reasoning Challenge dataset, which is a multiple-choice QA dataset with science exam questions from grades 3 to 9.

Table 2: Datasets used in our experiments.

Dataset	Size
AIME (Zhang et al., 2023a)	933
GSM8K (Cobbe et al., 2021)	1319
MATH-500 (Lightman et al., 2023)	500
ARC (Clark et al., 2018)	1172
HotpotQA (GLM, 2024)	100

4.2 Baselines

We evaluate CoM using zero-shot prompting, as few-shot approaches rely on task-specific examples, making them unsuitable for cross-domain comparisons. We compare CoM with three baselines that use recent prompting techniques and the same methodology definitions (Figure 8), as well as Macro-o1 (Zhao et al., 2024), a recent open-source model similar to OpenAI o1.

CoT (Wei et al., 2022) prompts the LLM to generate a chain of reasoning steps and shows the final result format.

MCoT provides the same methodology definitions as CoM, along with a CoT instruction to guide the LLM in using these methodologies in an appropriate order. MCoT evaluates whether methodologies can enhance reasoning in a single-turn, non-interactive setting, drawing on ideas from Least-to-Most (Zhou et al., 2023) and Metacognitive-Prompting (Wang and Zhao, 2024).

Both CoT and MCoT prompt the LLM once and do not allow code generation, as a second prompt is needed to synthesize code output.

Workflow is similar to CoM but uses a fixed methodology sequence per task, derived from the most frequent sequences chosen by CoM (Table 4). It guides the LLM through multiple reasoning turns, with sequences [Analysis, Coding, Variation, Conclusion] for AIME, GSM8K, and MATH, and [Analysis, Retrieval, Conclusion] for Hard Hotpot and ARC. Workflow incorporates ideas from Program-of-Thoughts (Chen et al., 2023), Cognitive Prompting (Wang and Zhao, 2024), workflow/pipeline (Jin et al., 2024; Anonymous, 2024b), and RAG (Anonymous, 2025).

Table 3: Performance of Macro-o1 and Qwen2.5-7B-Instruct on AIME Tasks and Hard Hotpot

	AIME	Hard Hotpot			
	Acc	**EM**	**F1**	**Prec**	**Rec**
Macro-o1					
CoT	14.47	0.12	0.20	0.20	0.27
MCoT	10.50	0.09	0.19	0.19	0.25
Qwen2.5-7B-Instruct					
CoT	20.15	0.17	0.25	0.25	0.24
MCoT	17.58	0.15	0.22	0.22	0.22
CoM	25.4	0.33	0.42	0.42	0.51

Table 4: Top 52.2% selected methodology sequences on AIME

	Methodology Sequence
22.0%	Analysis Coding Validation Conclusion
16.2%	Analysis Coding Conclusion
5.4%	Analysis Coding Validation Reflection Flexibility Conclusion
4.4%	Analysis Coding Validation Reflection Conclusion
4.3%	Analysis Coding Validation Reflection Flexibility Validation Conclusion

Table 5: Ablation Study Results for COM Method

CoM	**AIME (%)**	**Hard Hotpot**
No Ablation	25.4	0.4174
- Interpreter	14.1 (-44.5%)	0.25 (-40.2%)
- Analysis	18.7 (-26.6%)	0.38 (-8.4%)
- Coding	23.3 (-8.3%)	0.38 (-8.8%)
- Retrieval	-	0.22 (-46.8%)
- Validation	23.9 (-7.2%)	0.4 (-3%)
- Reflection	22.8 (-10.5%)	0.38 (-9%)
- Synthesis	23 (-9.3%)	0.4 (-3%)

4.3 Performance Comparison

Using the large LLM Qwen2-72B-Instruct, Figure 4 shows that CoM outperforms baselines on AIME and Hard Hotpot, with accuracy and F1 improvements of 38.5% and 28.7%, respectively, over CoT. Results are similar for DeepSeek-V3 in Figure 5. However, CoT slightly outperforms CoM on GSM8K, likely due to its simplicity and benchmark leakage (Xu et al., 2024b). Workflow, which is task-specifically optimized, ranks second, while MCoT results suggest minimal benefits from single-prompt methodologies.

As shown in Figure 4, when compared with the task-specifically optimized Workflow, CoM's accuracy is 1.7% higher on AIME, and 9.8% higher on Hard Hotpot, demonstrating CoM's superior flexibility in methodology selection. This highlights the effectiveness of metacognitive abilities in LLMs for choosing appropriate methodology sequences and validates our step-by-step reasoning approach.

With the smaller LLM Qwen2.5-7B-Instruct (Figure 6), CoM remains the best performer on AIME, MATH, and ARC. Likely due to benchmark leakage (Xu et al., 2024b), both CoM and Workflow show lower accuracy. On Hard Hotpot, CoM slightly underperforms Workflow, suggesting weaker metacognitive abilities in smaller models for methodology selection.

Finally, we compare CoM with Macro-o1 (Zhao et al., 2024) in Table 3. Results reveal that fine-tuning fails to improve Macro-o1 on AIME and Hard Hotpot, indicating insufficient generality in the fine-tuning data.

4.4 Methodology Selection Patterns

We analyzed the reasoning history from the experiment to identify the most frequent methodology sequences selected by CoM. Table 4 presents the top five patterns, which account for 52% of CoM's responses using Qwen2-72B-Instruct.

In 22.0% of cases, CoM followed a structured approach: analyzing, generating and executing code, validating results, and drawing conclusions. In 16.2% of cases, the model skipped validation, suggesting high confidence in its code. The remaining patterns involved additional error correction steps, indicating potential validation issues. These findings suggest that LLMs exhibit metacognitive abilities by planning their reasoning steps during problem-solving.

4.5 Ablation Study

We study the relative importance of each component of our CoM, including the Python interpreter, and each of the methodologies we used. Here, in contrast to excluding the *code* methodology, which prevents CoM from generating code, removing the Python interpreter still allows the LLM to generate code, but the LLM then needs to guess the output of the code by itself without an interpreter. Our ablation study is conducted with Qwen2.5-7B-Instruct.

As listed in Table 5, the interpreter is very important for both tasks, which shows that the code output guessed by the LLM without using the code interpreter for both math calculation and knowledge retrieval is unreliable. Secondly, for hard math problems in AIME, systematic analysis of the data and constraints in the problem is vital for the correctness of the reasoning. For AIME, all methodologies we provided are useful, each contributing to a 7-10% improvement in accuracy. In AIME, we disable retrieval for experimental simplicity. For Hard Hotpot, where reasoning relies on

Table 6: Average Speed of Experiments in Seconds per Iteration (Multiplied by 50)

	AIME	GSM8K	Hard Hotpot
Macro-o1			
CoT	96.0	33.5	19.0
MCoT	84.5	42.5	20.0
Qwen2.5-7B-Instruct			
CoM	91.0	34.0	50.5
Workflow	36.0	18.0	21.5
CoT	19.0	5.0	3.5
MCoT	19.0	8.0	3.5

Table 7: Average Number of Prompts

	AIME	GSM8K	Hard Hotpot
CoM	2×5.76	2×3.99	2×5.98
Workflow	4	4	3

retrieved information, retrieval is clearly the most important methodology.

4.6 Error Analysis

Errors made in CoM are conventional LLM errors such as hallucination, misunderstanding, and instruction-following errors. We manually inspected the first 10 error cases in CoM on the GSM8K dataset. We found that in most of these cases, methodology selection is not perfect. Three error cases are due to hallucination, where the wrong answers are given directly without the necessary calculation process. Two cases are due to translation errors from natural language to math; for example, "born early" is translated to a reduction in age. Three cases are due to language understanding errors; for instance, "restart downloading" is understood as "continue downloading", and "every second" is understood as "from the second". In one error case, the initial calculation is correct, but then a validation step causes an error because the LLM believes "servings" must be an integer. In one error case, the LLM generates more than one code block, although the methodology definition contains an instruction to generate a single standalone code block.

4.7 Efficiency

We examine the inference efficiency in terms of total inference time for speed and the number of inferences for cost. Table 6, shows that the speed of CoM is around 5 times that of CoT in AIME and 7 times in Hard Hotpot. However, CoM is comparable to the fine-tuned model, which generates longer reasoning traces with a lower speed per token.

Table 7 compares the number of prompts made by CoM with those made by Workflow. The results show that although we set the maximum iteration $K = 8$, CoM stops at a smaller number of steps on average, generates more reasoning steps for the harder AIME problems, and a smaller number of steps for the easier GSM8K problems.

4.8 Summary of Experiments

The experiments on complex mathematical problems (AIME and GSM8K) and multi-hop question answering (HotpotQA) evaluate the effectiveness of CoM in methodology selection and guided reasoning using a 72B, a 7B LLM, and a fine-tuned LLM for structured reasoning.

Results show that our CoM is effective in improving the performance of two challenging tasks over baselines that embody recent prompt engineering approaches. This result supports our hypothesis that we can use a training-free solution that integrates human methodological insights to enhance the performance of LLMs in complex reasoning.

Methodology selection patterns reveal that CoM effectively generates reasonable methodology sequences, which guide its reasoning in the right direction. Error analysis identifies that common LLM issues contribute to the majority of errors made by CoM. Finally, the ablation study confirms that the methodologies we employed are critical for solving complex reasoning tasks.

5 Conclusion and Future Work

This paper enhances LLMs' reasoning capabilities for complex tasks by simulating metacognitive processes and leveraging user-defined methodologies, enabling effective navigation of complex reasoning tasks without extensive retraining. Takeaways include: (1) LLMs exhibit latent metacognitive abilities that can be activated through structured, justification-driven prompting–eliminating the need for fine-tuning; and (2) generating explicit methodology justifications improves traceability and task comprehension, boosting zero-shot accuracy and cross-domain adaptation, which is often constrained by limited in-context examples.

A promising direction for future work is the automated search for optimal methodologies and reasoning chains, inspired by approaches like (Yang et al., 2024b), to identify and apply methodologies that enhance reasoning performance through better prompt construction.

Limitations

The approach proposed in this paper assumes that the LLM possesses metacognitive abilities. We found that other LLMs, despite demonstrating competitive performance in various benchmarks, fail in methodology selection, even with extensive prompt tuning efforts. For instance, one of these LLMs consistently selects the first methodology it initially chose. Additional experimental results illustrating these failures are provided in Appendix A.2.

Currently, our method requires two distinct prompts at each step: one for methodology selection and another for methodology-based reasoning. We attempted to consolidate these two prompts into a single one; however, we observed that even the most advanced LLMs we tested, including DeepSeek-V3, struggled to follow instructions with the combined, more complex prompt. We anticipate that future advancements in LLMs' instruction-following capabilities will enable the use of a single prompt, thereby improving the efficiency of our method.

The methodologies included in our framework are not exhaustive, leaving room for future research to expand and refine the list. Incorporating a wider range of strategies could enhance the adaptability and robustness of the CoM framework, opening new avenues for exploration and improvement.

Ethical Statement

This work fully complies with the ACL Ethics Policy. We declare that there are no ethical issues in this paper, to the best of our knowledge.

References

https://cloud.baidu.com/.

https://siliconflow.cn/.

Chao Li Chengen Huang Ge Zhang Guanwei Zhang Guoyin Wang Heng Li Jiangcheng Zhu Jianqun Chen Jing Chang Kaidong Yu Peng Liu Qiang Liu Shawn Yue Senbin Yang Shiming Yang Wen Xie Wenhao Huang Xiaohui Hu Xiaoyi Ren Xinyao Niu Pengcheng Nie Yanpeng Li Yuchi Xu Yudong Liu Yue Wang Yuxuan Cai Zhenyu Gu Zhiyuan Liu Zonghong Dai 01.AI: Alex Young, Bei Chen. 2024. Yi: Open Foundation Models by 01.AI. *Preprint*, arXiv:2403.04652.

Anonymous. 2024a. AFlow: Automating agentic workflow generation. In *The Thirteenth International Conference on Learning Representations (ICLR)*.

Anonymous. 2024b. Chain of ideas: Revolutionizing research in idea development with LLM agents. In *The Thirteenth International Conference on Learning Representations (ICLR)*.

Anonymous. 2025. Inference scaling for long-context retrieval augmented generation. In *The Thirteenth International Conference on Learning Representations (ICLR)*.

Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, Xiaoyi Dong, Haodong Duan, Qi Fan, Zhaoye Fei, Yang Gao, Jiaye Ge, Chenya Gu, Yuzhe Gu, Tao Gui, Aijia Guo, Qipeng Guo, Conghui He, Yingfan Hu, Ting Huang, Tao Jiang, Penglong Jiao, Zhenjiang Jin, Zhikai Lei, Jiaxing Li, Jingwen Li, Linyang Li, Shuaibin Li, Wei Li, Yining Li, Hongwei Liu, Jiangning Liu, Jiawei Hong, Kaiwen Liu, Kuikun Liu, Xiaoran Liu, Chengqi Lv, Haijun Lv, Kai Lv, Li Ma, Runyuan Ma, Zerun Ma, Wenchang Ning, Linke Ouyang, Jiantao Qiu, Yuan Qu, Fukai Shang, Yunfan Shao, Demin Song, Zifan Song, Zhihao Sui, Peng Sun, Yu Sun, Huanze Tang, Bin Wang, Guoteng Wang, Jiaqi Wang, Jiayu Wang, Rui Wang, Yudong Wang, Ziyi Wang, Xingjian Wei, Qizhen Weng, Fan Wu, Yingtong Xiong, Chao Xu, Ruiliang Xu, Hang Yan, Yirong Yan, Xiaogui Yang, Haochen Ye, Huaiyuan Ying, Jia Yu, Jing Yu, Yuhang Zang, Chuyu Zhang, Li Zhang, Pan Zhang, Peng Zhang, Ruijie Zhang, Shuo Zhang, Songyang Zhang, Wenjian Zhang, Wenwei Zhang, Xingcheng Zhang, Xinyue Zhang, Hui Zhao, Qian Zhao, Xiaomeng Zhao, Fengzhe Zhou, Zaida Zhou, Jingming Zhuo, Yicheng Zou, Xipeng Qiu, Yu Qiao, and Dahua Lin. 2024. Internlm2 technical report. *Preprint*, arXiv:2403.17297.

Shulin Cao, Jiajie Zhang, Jiaxin Shi, Xin Lv, Zijun Yao, Qi Tian, Lei Hou, and Juanzi Li. 2023. Probabilistic tree-of-thought reasoning for answering knowledge-intensive complex questions. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12541–12560, Singapore. Association for Computational Linguistics.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*.

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

DeepSeek-AI. 2024. Deepseek-v3 technical report. https://arxiv.org/pdf/2412.19437.

Aniket Rajiv Didolkar, Anirudh Goyal, Nan Rosemary Ke, Siyuan Guo, Michal Valko, Timothy P Lillicrap, Danilo Jimenez Rezende, Yoshua Bengio, Michael Curtis Mozer, and Sanjeev Arora. 2024. Metacognitive capabilities of LLMs: An exploration in mathematical problem solving. In *AI for Math Workshop @ ICML 2024*.

Marah Abdin et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *Preprint*, arXiv:2404.14219.

Team GLM. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *Preprint*, arXiv:2406.12793.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.

Mingyu Jin, Weidi Luo, Sitao Cheng, Xinyi Wang, Wenyue Hua, Ruixiang Tang, William Yang Wang, and Yongfeng Zhang. 2024. Disentangling memory and reasoning ability in large language models. *Preprint*, arXiv:2411.13504.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *ICLR 2024*.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 305–329, Nusa Dua, Bali. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *NeurIPS*.

OpenAI. 2024. Learning to Reason with LLMs.

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

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *Preprint*, arXiv:2408.03314.

O. Team. 2024. https://github.com/open-source-o1/open-o1.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

Boshi Wang, Xiang Deng, and Huan Sun. 2022. Iteratively prompt pre-trained language models for chain of thought. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2714–2730, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jianing Wang, Qiushi Sun, Xiang Li, and Ming Gao. 2024. Boosting language models reasoning with chain-of-knowledge prompting. In *The 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4958–4981, Bangkok, Thailand. Association for Computational Linguistics.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023a. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2609–2634, Toronto, Canada. Association for Computational Linguistics.

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*.

Yuqing Wang and Yun Zhao. 2024. Metacognitive prompting improves understanding in 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 1914–1926, Mexico City, Mexico. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Jundong Xu, Hao Fei, Liangming Pan, Qian Liu, Mong-Li Lee, and Wynne Hsu. 2024a. Faithful logical

reasoning via symbolic chain-of-thought. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13326–13365, Bangkok, Thailand. Association for Computational Linguistics.

Ruijie Xu, Zengzhi Wang, Run-Ze Fan, and Pengfei Liu. 2024b. Benchmarking benchmark leakage in large language models. *arXiv preprint arXiv:2404.18824*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2024b. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*.

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. *arXiv preprint arXiv:2406.04271*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, volume 35, pages 15476–15488. Curran Associates, Inc.

Dan Zhang, Sining Zhoubian, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024. Rest-mcts*: Llm self-training via process reward guided tree search. Thirty-eighth Conference on Neural Information Processing Systems (NeurIPS).

Xingyuan Zhang, Philip Becker-Ehmck, Patrick van der Smagt, and Maximilian Karl. 2023a. Action inference by maximising evidence: Zero-shot imitation from observation with world models. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*.

Zhebin Zhang, Xinyu Zhang, Yuanhang Ren, Saijiang Shi, Meng Han, Yongkang Wu, Ruofei Lai, and Zhao Cao. 2023b. IAG: Induction-augmented generation framework for answering reasoning questions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1–14, Singapore. Association for Computational Linguistics.

Yu Zhao, Huifeng Yin, Bo Zeng, Hao Wang, Tianqi Shi, Chenyang Lyu, Longyue Wang, Weihua Luo, and Kaifu Zhang. 2024. Marco-o1: Towards open reasoning models for open-ended solutions. *Preprint*, arXiv:2411.14405.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

A Appendix

A.1 Prompts

Our list of methodologies is displayed in Figure 8, and our methodology-selection and methodology-based reasoning prompts are listed in Figure 7.

A.2 Additional Experiment Results

The approach proposed in this paper assumes that the LLM possesses metacognitive abilities. This section presents additional experimental results in Figures 9, 10, and 11, which reveal that some LLMs, despite demonstrating competitive performance across various benchmarks, struggle with methodology selection even after extensive prompt tuning efforts. For example, one of these LLMs consistently defaults to selecting the first methodology it initially identifies, highlighting a limitation in its decision-making process.

 # List of Existing Methodologies

{example_methodologies}

{historical_reasoning_steps}

 # Question
- {question}

 # Step Instruction
- You are currently on {cur_step} out of a total of a maximum of {max_steps} steps to solve the question. Please select the most suitable methodology from the list, considering the reasoning history and the timing for applying each methodology. Try to diversify your choices to advance the resolution of the question, and avoid redundant or repeated methodologies. Select and output a single methodology succinctly in the following format, without any additional text:

[Methodology name]
- When: [The timing for applying the methodology]
- What: [The characteristics and details of the methodology]

 # List of Existing Methodologies

{example_methodologies}

{historical_reasoning_steps}

 # Question
- {question}

 # Selected Methodology

{selected_methodology}

 # Step Instruction
- You are currently on {cur_step} out of a total of a maximum of {max_steps} steps to solve the question. Please conduct an innovative next reasoning step for the question, using the selected methodology "{selected_methodology_name}" based on the current reasoning history. Please output your reasoning step succinctly in the following format, without any additional text:

- Methodology: [Name of the selected methodology]
- Reasoning/Code: [A structural and systematic reasoning step that uses the selected methodology and improves on your past reasoning steps. Or a Python snippet with such reasoning inside its comments in a ```python\n...``` code-block.]
- Result: [A short result of your reasoning, or an accurate simulated outpout from the code.]

Figure 7: Our methodology-selection prompt (left) and methodology-based reasoning prompt (right).

Analysis
- When: In step 1.
- What: Analyze the category and solution type of the question, list the key facts, variables, relations, constraints with their associated values, and clarify the required output format. Break down complex problems into simpler steps while maintaining critical context. Propose a sequence of methodologies necessary to tackle the remaining reasoning steps iteratively and explain how they are related to the final result.

Retrieval
- When: Fact-based information from the internet is needed.
- What: Write 1-3 line(s) of Python function call(s) `search([information],topk=3)` for each information needed to retrieve. The function `search` has been defined and imported for you, which returns a text summary for the argument `information`. Place your code in a single ```python\n...``` code-block. Finally, accurately simulated the retrieved output by yourself.

Coding
- When: Coding is necessary.
- What: A standalone Python snippet with structural and systematic reasoning in comments, using the **print** function to output the result. Place your code in a single ```python\n...``` code-block. Finally, accurately simulated the output of your Python snippet by yourself without using a computer.

Validation
- When: A temporary or a final the result is resulting from a previous reasoning step.
- What: Identify the result, and analyze its correctness from a different angle. You may write a test-case function that prints True/False to validate the result and then simulate its output.

Reflection
- When: An error is detected or validation fails.
- What: Analyze the reasoning steps to identify errors and provide constructive self-critic or feedback for improvement.

Flexibility
- When: The previous step fails to obtains the expected result or when a reflective or critic feedback is available.
- What: Adjust the approach based on insights gained and propose alternative strategies for the next steps.

Conclusion
- When: A confident final answer is available, or in the last step.
- What: Clarify the output format required by the question. Compile the reasoning process and generate the answer in the required format.

Figure 8: Our list of methodologies.

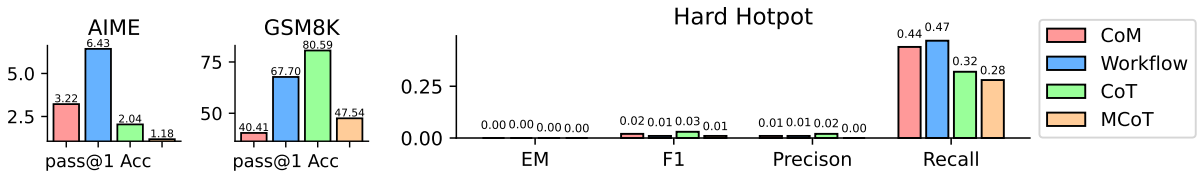

Figure 9: Results of Yi-1.5-9B-Chat (01.AI: Alex Young, 2024) on AIME, GSM8K, and Hard HotpotQA.

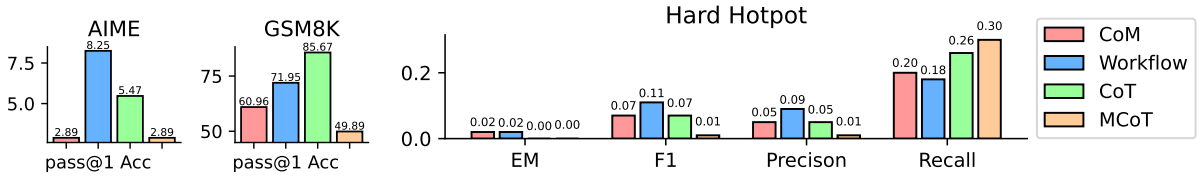

Figure 10: Results of InternLM2.5-7B-chat (Cai et al., 2024) on AIME, GSM8K, and Hard HotpotQA.

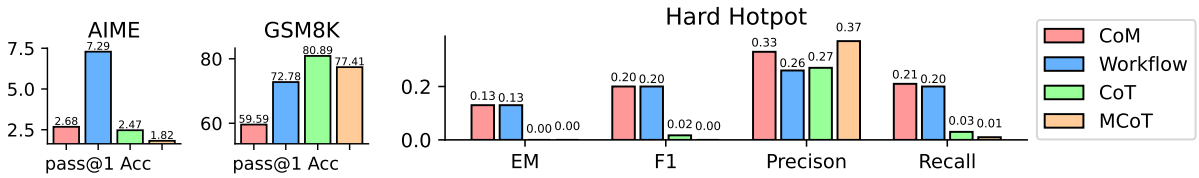

Figure 11: Results of GLM-4-9b-chat (GLM, 2024) on AIME, GSM8K, and Hard HotpotQA.