Reason from Future: Reverse Thought Chain Enhances LLM Reasoning

Yinlong Xu[1], **Yanzhao Zheng**[2], **Shuoshuo Sun**[2],
Shuaihan Huang[2], Baohua Dong[2], Hangcheng Zhu[2], Ruohui Huang[2], Gang Yu[2], Hongxia Xu[3,4*], Jian Wu[1,5*]
[1] College of Computer Science and Technology, Zhejiang University, Hangzhou, China
[2] Alibaba Group, Hangzhou, China
[3] State Key Laboratory of Transvascular Implantation Devices and TIDRI, Hangzhou, 310009, China
[4] Liangzhu Laboratory and WeDoctor Cloud, Hangzhou, 310000, China
[5] Zhejiang Key Laboratory of Medical Imaging Artificial Intelligence, Hangzhou, 310058, China
{xuyinlong, Einstein, Wujian2000}@zju.edu.cn huangshuaihan@outlook.com
{zhengyanzhao.zyz, sunshuoshuo.sss, baohua.dbh, linran.lr09, wentong, ruohai}@taobao.com

Abstract

It has been demonstrated that carefully designed reasoning paradigms, like Chain-of-Thought (CoT) and Tree-of-Thought (ToT), can enhance the reasoning capabilities of small language models by detailed thinking and extensive thought searching, unbounded branching factors in the searching space create prohibitive reasoning consumption. However these methods fall into the trap of local optimum reasoning, which means the model lacks a global perspective while solving problems. We propose a novel reasoning paradigm called *Reason from Future* (RFF), which generates reasoning paths by bidirectional reasoning that combines top-down planning with bottom-up reasoning accumulation. The essence of RFF lies in its reverse reasoning mechanism, which prioritizes core logical relationships and imposes goal-oriented constraints on intermediate steps, thereby reducing the searching space and mitigating error accumulation inherent in sequential forward reasoning. Empirical evaluations across diverse experiments demonstrate that RFF outperforms conventional paradigms with higher accuracy and less searching space to solve complex tasks.

1 Introduction

The rapid evolution of large language models (LLMs), fueled by breakthroughs in deep learning architectures and unprecedented datasets, has demonstrated remarkable potential across natural language processing (NLP) and interdisciplinary applications (Lee and Toutanova, 2018; Radford, 2018; Team et al., 2023; Sel et al., 2023). LLMs like ChatGPT (Achiam et al., 2023) and Llama (Dubey et al., 2024) exhibit human-like text generation, multilingual task execution, and emerging logical reasoning. Current scholarly investigations identify their reasoning capacity for problem decomposition as the critical determinant of functional boundaries, enabling industrial automation and academic research applications.

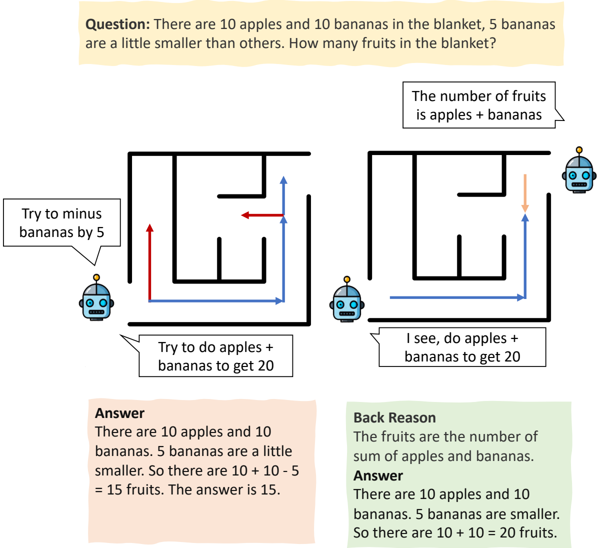

Question: There are 10 apples and 10 bananas in the blanket, 5 bananas are a little smaller than others. How many fruits in the blanket?
The number of fruits is apples + bananas
Try to minus bananas by 5
Try to do apples + bananas to get 20
I see, do apples + bananas to get 20
Answer
There are 10 apples and 10 bananas. 5 bananas are a little smaller. So there are 10 + 10 - 5 = 15 fruits. The answer is 15.
Back Reason
The fruits are the number of sum of apples and bananas.
Answer
There are 10 apples and 10 bananas. 5 bananas are smaller. So there are 10 + 10 = 20 fruits.

Figure 1: Comparison between simple forward reasoning (left) and forward reasoning guided by back reasoning (right).

Recent studies demonstrate that well-designed reasoning paradigms can significantly enhance LLMs' reasoning ability without additional costly and time-consuming post-training. A seminal work in this area is Chain-of-Thought (CoT) (Wei et al., 2022), which pioneered the novel view that reasoning ability can be improved by designing reasoning prompts, paradigms, and examples. Tree-of-Thought (ToT) (Yao et al., 2024) provides a searching view to enhance the ability of complex reasoning. Progressive-Hint Prompting (PHP) (Zheng et al., 2023) and Cumulative Reasoning (CR) (Zhang et al., 2023) asks the model to generate hints for the question before generating the answer.

Although, these reasoning paradigms, break-

[*] Corresponding authors.

ing down the solution into multiple steps through prompts or spatial search, can enhance the reasoning ability and coherence of the model. They tend to make the model focus on the current state, resulting in lacking explicit guidance from a global understanding of the problem and excessive exploration of redundant information, overthinking, or errors during inference (Boix-Adsera et al., 2023).

In contrast, the way human approaches problem-solving is different. Researches have shown that humans begin by building holistic mental modeling when solving complex problems, allowing problem solvers to form a topological framework before focusing on specific details (Spreng et al., 2009; Koban et al., 2021). This kind of cognitive prediction provides dual guidance for the subsequent solution process: forming a "cognitive road map" of the solution path at the macro-level, helping to exclude obviously unrelated branches; evaluation criteria are established at the micro-level so that each specific operation remains dynamically calibrated to the end goal. This global awareness allows us to avoid blindly combining superficial details and instead prioritize purposeful, contextually grounded deductions. This suggests that modeling this local-global consistency thinking paradigm might be able to enable LLMs to strategically synthesize information, minimize irrelevant exploration, and align intermediate steps with the overarching goal.

Inspired by the maze-solving strategy of backward reasoning, where reversing the path from the endpoint accelerates discovering the solution, we propose a novel reasoning paradigm called **R**eason-**f**rom-**F**uture (RFF) to enhance the reasoning ability of LLMs by adding reverse thinking process to guide the forward reasoning as shown in Figure 1.

RFF integrates bidirectional reasoning by alternating between reverse and forward thinking to maintain solution states: the reverse reasoning generates the potential last state of the target state and sets the last state as the new target, then the forward reasoning takes a step toward the new target. The target state serves as a guide to precisely lead the forward reasoning, and the forward reasoning in turn produces more useful information to make the reverse reasoning more reasonable. We evaluate RFF in five datasets: Game of 24 (Yao et al., 2024), GSM8K (Cobbe et al., 2021), ASDiv (Miao et al., 2021), SVAMP (Patel et al., 2021) MATH-500 (Lightman et al., 2023), and demonstrate significant improvements in accuracy over baseline methods. Additionally, RFF reduces the search space by constraining reasoning to target-driven states, demonstrating good efficiency. Our results highlight the potential of bidirectional, goal-aware reasoning to unlock more robust and systematic problem-solving in LLMs.

In summary, We introduce RFF, a novel self-planning reasoning paradigm to enhance the reason ability of LLMs. In which, reverse thinking and forward-thinking alternately to obtain a future perspective and narrow the solution-searching space. We conduct experiments involving four datasets to demonstrate the great performance and efficiency of RFF. And we employ two extra experiments by complicating the questions in Game of 24 and GSM8K. The results represent RFF less consuming in larger search spaces and robust thinking in variant problems.

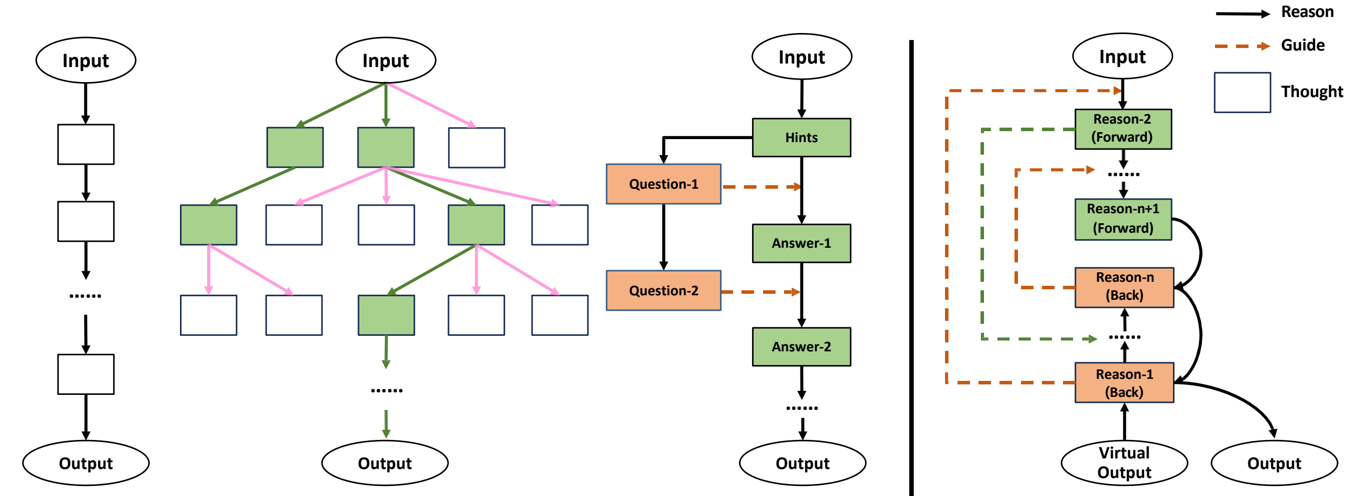
2 Related Work

2.1 Chain of Thought Reasoning

In the study of complex reasoning tasks, Chain-of-Thought (CoT) (Wei et al., 2022; Wang et al., 2022) prompting has emerged as a pivotal technique for significantly improving the performance of large language models (LLMs) by explicitly generating intermediate reasoning steps. This approach enables the decomposition of problems into structured, stepwise reasoning pathways, demonstrating particular efficacy in mathematical and logical domains. Recent advancements extend CoT through symbolic formalization (e.g., Symbolic CoT (Xu et al., 2024)), which incorporates formal logic systems to enhance both reliability and interpretability by grounding reasoning in rigorous symbolic frameworks. Critical analyses, however, reveal potential limitations where models may exploit computational redundancy rather than genuine reasoning in extended CoT steps, prompting discussions about mechanistic transparency.

2.2 Search Reasoning

In the domain of search-based reasoning for large language models, the Tree-of-Thought (ToT) (Yao et al., 2024) framework introduces backtracking capabilities within multi-path decision structures, enabling systematic exploration of diverse solution trajectories. This approach proves particularly effective for complex tasks requiring iterative hypothesis generation and validation. Monte Carlo Tree Search (MCTS) (Świechowski et al., 2023) strengthens online decision-making robustness



(a) Chain-of-Thought Prompting (COT)

(b) Tree-of-Thought Prompting (TOT)

(c) Cumulate Reasoning Prompting (CR)

(d) Reason from Future Prompting (RFF)

Figure 2: Schematic illustrating various approaches to problem-solving with LLMs, and each rectangle box represents a thought. Figure2(d) only shows the basic framework about RFF, see the concrete pipeline of two types of RFF in Algorithm 1 (RFF-T) and Algorithm 2 (RFF-G).

through simulating and evaluating long-term rewards of candidate paths, demonstrating strengths in reinforcement learning and dynamic programming scenarios. AOT (Sel et al., 2023) improves TOT by introducing a step-evaluation mechanism into the reasoning process, which helps LLM to prune the less likely searching route thus reduce the searching space. AOT+ (Sel et al., 2025), as the upgrade of AOT, adds a fine-grained backtracking mechanism by labeling each steps, which further reduces the reasoning consumption on an error searching route. However, both AOT and AOT+ get their global perspectives by continue exploring reasoning and degrade to normal TOT when exploring their first reasoning route, which may lead to a random search in each step and may miss the correct searching route.

Recent innovations also bridge reasoning with executable action, exemplified by frameworks like LATS (Language Agent Tree Search) (Zhou et al., 2023). By unifying hierarchical planning, probabilistic reasoning, and environment interaction within language models, LATS extends the dynamic capabilities of ReAct (Reasoning + Acting) (Yao et al., 2022) paradigms, enabling adaptive agent behavior in multi-step problem-solving scenarios. While these approaches show complementary advantages in addressing combinatorial optimization and long-range dependency challenges, computational efficiency and path-pruning strategies remain critical areas for improvement.

2.3 Progressive Hint Prompting Reasoning

In the realm of progressive prompting for complex reasoning, the LLMs solve a problem through multiple rounds of messages. Least-to-Most (Zhou et al., 2022) first breaks a problem into several sub-problems and then solve them sequentially. Progressive-Hint Prompting (PHP) (Zheng et al., 2023) advances dynamic problem-solving by fostering iterative, multi-turn interactions between users and LLMs. This method leverages feedback-driven prompts informed by historical outputs to systematically refine reasoning accuracy and coherence. Parallel to this, Cumulative Reasoning (CR) (Zhang et al., 2023) emulates human-like incremental cognition by decomposing tasks into structured subtasks and aggregating intermediate results through stepwise integration. Both PHP and CR synergize with foundational frameworks like CoT and its derivatives, collectively strengthening the generation and validation of adaptive reasoning pathways.

Recent advancements further explore hybrid architectures that combine PHP with retrieval-augmented mechanisms and task-specific distillation. These frameworks aim to balance computational efficiency with robust reasoning fidelity, ad-

Algorithm 1 RFF-T

```
Require: LM $p_\theta$, input $x$, max Steps $L$, last step generator $G()$, stepwise reasoner $R()$, state checker $C()$, current state $\{S\}$, target state $\{T\}$, avoid attempts $\{A\}$, verifier $V()$, Output function $O()$
$S_0 \leftarrow x, T_0 \leftarrow t, A_0 \leftarrow \{\}, i \leftarrow 0$
while $i <= L$ do
    $i \leftarrow i + 1$
    $A_i \leftarrow \{\}$
    $T_i \leftarrow G(p_\theta, S_{i-1}, T_{i-1})$
    $S_i \leftarrow R(p_\theta, S_{i-1}, T_i, A_{i-1})$
    if $C(S_i, T_i) == True$ then
        $j \leftarrow V(S_i, T_i)$
        if $j == i$ then
            break
        end if
        $A_j \leftarrow A_j \cup \{S_j, T_j\}$
        $i \leftarrow j$
    end if
end while
return $O(p_\theta, x, t|S_i)$
```

dressing challenges such as error propagation and context scalability. By integrating iterative feedback loops with external knowledge retrieval, such approaches optimize performance in multi-step reasoning tasks while maintaining generalizability.

3 Methods

Reason from Future (RFF) is a reasoning paradigm that allows models to solve a question by using forward and backward reasoning alternately. We use p_θ to denote a LLM with parameters p_θ, and x, t to denote the $input$ and $question$. The $\{S\} \sim \{S_0, S_1 ... S_i\}, \{T\} \sim \{T_0, T_1 ... T_i\}$ denote the list of current states and the list of target states in each step i. We define $O(p_\theta, x, t|S_i)$ as the output of the LLM with parameters p_θ using a prompt consisting of input x, target t, and hints S_i. In the $i - th$ step, the model identifies the preceding step closest to the current target state T_{i-1} and considers it as the new target state T_i and provides the calculation relationship between the two. Then the model takes the T_{i-1} as the new target for one-step forward reasoning. The model then repeats this step until the latest target state has been achieved ($S_i = T_i$). A specific RFF pipeline should consist of three components: 1: Last Step Generator $G()$; 2: Stepwise Forward Reason $R()$; 3: State Check $C()$.

3.1 Last Step Generator

RFF implements backward reasoning by generating the last previous step. To be specific, RFF decomposes one target state T_i with current state S_i into a pre-target state $T_{i+1} = G(p_\theta, S_i, T_i)$ at a time, the form of the specific sub-target state depends on the on the target of the task, such as a set of numbers (Game of 24), the variables to be found (mathematical problems). It is worth noticing that the transition step between pre-target state T_{i+1} to target T_i should be output explicitly to guarantee the correctness of the target decomposition to a certain extent.

3.2 Stepwise Forward Reason

We consider two different strategies: RFF-T in Algorithm 1 and RFF-G in Algorithm 2, to generate the next forward reasoning step for different types of target:

(a) RFF-T: For problems like Game of 24 or Maze game, whose solution is one branch of a searching tree, the model should avoid repeating the wrong attempts in the same layer of the searching tree. We use $\{A\} \sim \{A_0, A_1 ... A_i\}$ to denote the attempts should be avoid in step i, thus the next state should be $S_i \leftarrow R(p_\theta, S_{i-1}, T_i, A_{i-1})$.

(b) RFF-G: For the problem like mathematical problems, whose solution is a directed acyclic graph, all the information calculated by the previous states are either useful or redundant but not harmful, so the reasoning path should consider all the information calculated by the previous states, which is $S_i \leftarrow S_{i-1} \cup R(p_\theta, x, S_{i-1}, T_i)$.

Algorithm 2 RFF-G

```
Require: LM $p_\theta$, input $x$, max Steps $L$, last step generator $G()$, stepwise reasoner $R()$, state checker $V()$, current state $\{S\}$, target state $\{T\}$, Output function $O()$
$S_0 \leftarrow x, T_0 \leftarrow t_0$
for $i = 1$ to $L$ do
    $T_i \leftarrow G(p_\theta, S_{i-1}, T_{i-1})$
    $S_i \leftarrow S_{i-1} \cup R(p_\theta, S_{i-1}, T_i)$
    if $V(S_i, T_i) == True$ then
        break
    end if
end for
return $O(p_\theta, x, t|S_i)$
```

3.3 State Check

State Check $C()$ maintains an inference boundary that determines the termination conditions of the inference paradigm. Similar to Stepwise Forward Reason, we set two different strategies to check whether the reasoning comes to the boundary:

(a) RFF-T: For the reason only the correct reasoning path will be saved in the end, the $C(p_\theta, S_i, T_i)$ only considers whether the current state S_i coincides with the latest target state T_i, or whether the current state requires only one mathematical or logical operation to reach the target state(e.g. present state:(2 3 4), target state:(4 6) in Game of 24). Meanwhile, because RFF-T need to revisit the previous state to explore the thought space, a Verifier $V(S_i, T_i)$ is set to verify whether this path is the correct path when the reasoning comes to the boundary. If the path is a wrong path $V()$ will return the previous state j which should be revisited and record the wrong attempt (S_j, T_j).

(b) RFF-G: Different from the RFF-T, each step of reasoning generates a useful node of the directed acyclic graph, so $C(p_\theta, S_i, T_i)$ considers whether the information the target state needs has already been solved or is noted in the background.

4 Experiment

We evaluate the effectiveness of RFF on some widely used LLM reasoning benchmarks, like GAME of 24 and GSM8k. Considering that successful paradigm may be due to the strength of the model itself rather than the strength of the paradigm, leading to difficulty in migrating them to weak models or small models, we carry out our experiments using Llama3-8B-Instruct (Dubey et al., 2024) and Qwen2.5-7B-Instruct (Yang et al., 2024) as the base models, and more detailed parameters will be shown in specific tasks setup.

4.1 Game of 24

The task of Game of 24 originates from (Yao et al., 2024), where the goal is to use four numbers with basic arithmetic operations (+-*/) to obtain 24, and each number can be used only once.

Task Setup

We conduct Game of 24 on Llama3-8B-Instruct with a temperature of 0.7 (consistent with the setup of CoT (Wei et al., 2022) and ToT (Yao et al., 2024)). We apply RFF-T because Game of 24 is usually viewed as fetching a branch of the searching tree and is consistent with the paradigm of RFF-T. A solving example can be seen in Figure 3. We conduct 100 times about the middle hard 100 puzzles from 901 to 1000 (Yao et al., 2024). We also consider each branch of the searching tree as a visit state and record the average visit states of different paradigms, which is proportional to the search space and the consumption of computation.

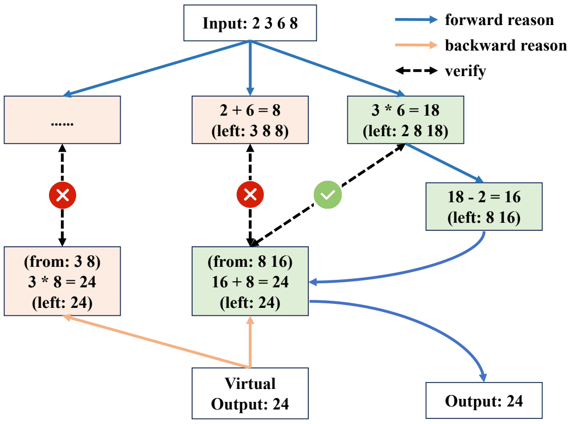


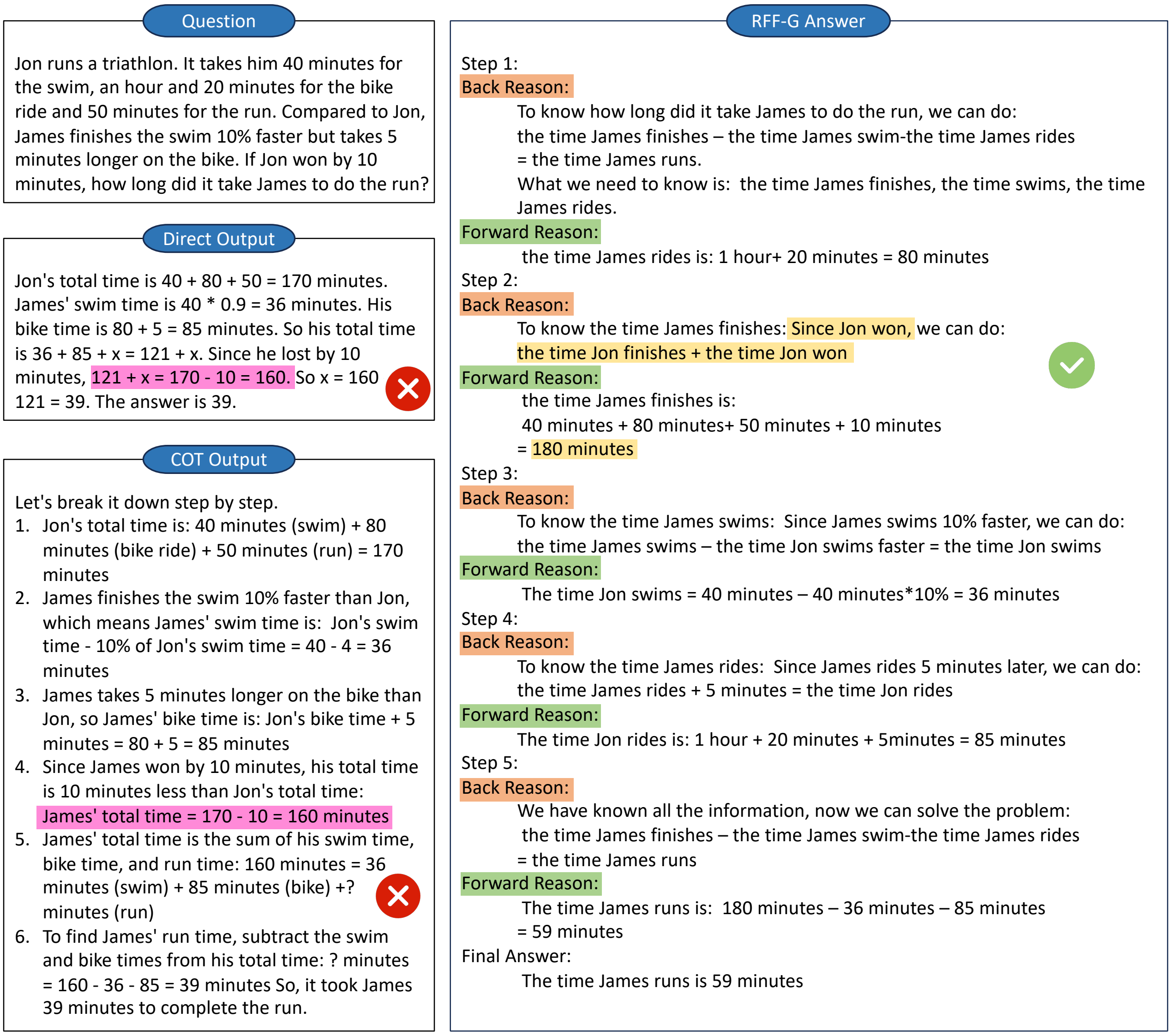
Figure 3: An example of how RFF-T works in Game of 24. Different reasoning paths represent the backtracking mechanism of RFF-T

Model	Method	ACC	Visited States
GPT-4	CoT	3%	1.0
	ToT(n=1)	45%	-
	ToT(n=5)	74%	61.2
	AoT	71%*	-
	CR(n=1)	84%	11.7
	CR(n=5)	94%	13.7
	RFF(n=5)	95%	9.3
Llama3-8B	CR(n=1)	9%	30.9
	CR(n=5)	19%	89.8
Llama3-8B	**RFF**(n=5)	89%	**9.9**
	RFF(n=10)	**96%**	15.0

Table 1: The results of the Game of 24, where n denotes the width of the searching tree. * denotes the result is from (Sel et al., 2023).

Baselines

We employ CoT, ToT, AoT and cumulative reasoning (CR) with different parameters as the baselines. The setup of CoT is consistent with (Wei et al., 2022) and (Yang et al., 2024), who employ the intermediate calculation process as the reasoning step. As for ToT and CR, we adapt the settings and



Question

Jon runs a triathlon. It takes him 40 minutes for the swim, an hour and 20 minutes for the bike ride and 50 minutes for the run. Compared to Jon, James finishes the swim 10% faster but takes 5 minutes longer on the bike. If Jon won by 10 minutes, how long did it take James to do the run?

Direct Output

Jon's total time is 40 + 80 + 50 = 170 minutes. James' swim time is 40 * 0.9 = 36 minutes. His bike time is 80 + 5 = 85 minutes. So his total time is 36 + 85 + x = 121 + x. Since he lost by 10 minutes, 121 + x = 170 - 10 = 160. So x = 160 121 = 39. The answer is 39. ✗

COT Output

Let's break it down step by step.

1. Jon's total time is: 40 minutes (swim) + 80 minutes (bike ride) + 50 minutes (run) = 170 minutes
2. James finishes the swim 10% faster than Jon, which means James' swim time is: Jon's swim time - 10% of Jon's swim time = 40 - 4 = 36 minutes
3. James takes 5 minutes longer on the bike than Jon, so James' bike time is: Jon's bike time + 5 minutes = 80 + 5 = 85 minutes
4. Since James won by 10 minutes, his total time is 10 minutes less than Jon's total time: James' total time = 170 - 10 = 160 minutes
5. James' total time is the sum of his swim time, bike time, and run time: 160 minutes = 36 minutes (swim) + 85 minutes (bike) +? minutes (run) ✗
6. To find James' run time, subtract the swim and bike times from his total time: ? minutes = 160 - 36 - 85 = 39 minutes So, it took James 39 minutes to complete the run.

RFF-G Answer

Step 1:
Back Reason:
To know how long did it take James to do the run, we can do:
the time James finishes – the time James swim-the time James rides = the time James runs.
What we need to know is: the time James finishes, the time swims, the time James rides.
Forward Reason:
the time James rides is: 1 hour+ 20 minutes = 80 minutes
Step 2:
Back Reason:
To know the time James finishes: Since Jon won, we can do:
the time Jon finishes + the time Jon won ✓
Forward Reason:
the time James finishes is:
40 minutes + 80 minutes+ 50 minutes + 10 minutes
= 180 minutes
Step 3:
Back Reason:
To know the time James swims: Since James swims 10% faster, we can do:
the time James swims – the time Jon swims faster = the time Jon swims
Forward Reason:
The time Jon swims = 40 minutes – 40 minutes*10% = 36 minutes
Step 4:
Back Reason:
To know the time James rides: Since James rides 5 minutes later, we can do:
the time James rides + 5 minutes = the time Jon rides
Forward Reason:
The time Jon rides is: 1 hour + 20 minutes + 5minutes = 85 minutes
Step 5:
Back Reason:
We have known all the information, now we can solve the problem:
the time James finishes – the time James swim-the time James rides = the time James runs
Forward Reason:
The time James runs is: 180 minutes – 36 minutes – 85 minutes = 59 minutes
Final Answer:
The time James runs is 59 minutes

Figure 4: An example from the GSM8K dataset, with solution generated by Direct, CoT, and RFF paradigms. The former two paradigms tend to connect with "win" to positive operation "more", while RFF will first analyze the background of the "win" and then generate the operation.

prompts from (Zhang et al., 2023). All methods are tested 100 times to get the average result, and unless otherwise specified, the temperature of the model is set to 0.7. We also use GPT-4 as the baseline model. Due to the different space exploring paradigms, we treat those with a similar number of search spaces as a same class for comparison instead of similar branches of searching trees.

Results

As shown in Table 1, RFF with Llama3-8B exhibits outstanding performance even compared to GPT-4. CoT performs badly on this task for the searching-tree-like tasks need the model to explore wide solution space. Searching paradigms like ToT and CR achieve better scores than CoT, meanwhile, the ToT method visits more states because of blind searching. RFF reaches the highest accuracy and least visit-states at the same level: when the visit-state is around 10, RFF reaches the best accuracy of 89% compared to CR with GPT-4 at 84%; when the visit-state is around 14, RFF reaches an accuracy of 96% compared to CR with GPT-4 at 94%. The fewer visit-states and higher accuracy are due to the searching space in RFF being much smaller and reasonable than simply forward searching (e.g. for "1 2 12 12", with the target "12+12=24", LLM will less likely explore ways like "2+12=14").

Model	Method	GSM8K	SVAMP	ASDiv	MATH	AVG
Llama3-8B-Instruct	CoT	75.6%	80.5%	82.3%	32.8%	67.8%
	Least-to-Most	79.5%	86.8%	84.4%	38.8%	72.4%
	Give-me-Hint	77.3%	87.9%	86.0%	37.0%	72.1%
	CR	77.0%	71.2%	84.8%	40.2%	68.3%
	RFF	**83.8%**	**89.7%**	**86.7%**	**41.4%**	**75.4%**
Qwen2.5-7B-Instruct	CoT	87.2%	92.1%	88.0%	74.6%	85.5%
	CR	87.7%	83.7%	91.9%	78.2%	85.4%
	RFF	**89.5%**	**95.1%**	**92.2%**	**79.8%**	**89.1%**

Table 2: The results of the math problems. The score represents the accuracy of the benchmarks. The best results in each column is highlighted in **bold**. The AVG represents the average value of the four benchmarks.

4.2 Math Problem Benchmark

This task contains four datasets: GSM8K, SVAMP, AsDiv and MATH-500. GSM8K is a mathematical dataset with 1319 test data, each question requires 3-10 steps of reasoning and calculation. SVAMP and ASDiv are two simple math problem datasets with 1000 and 2096 data respectively, each question requires 1-2 steps of reasoning and calculation. MATH-500 is a subset of 500 mathematical problem from the MATH (Hendrycks et al., 2021) benchmark, which is a much more harder benchmark than GSM8K.

Task Setup

We conduct this task on Llama3-8B-Instruct and Qwen2.5-7B-Instruct with a greedy search to exclude the influence of random numbers on textual reasoning. We apply RFF-G for the mathematical puzzle as its solution can be seen as a directed acyclic graph from the question to the answer. We employ 1 shot as the example to lead the model to perform formatted reasoning.

Baselines

Considering the nature of the math problems, CoT and CR are chosen as baselines for their excellent ability for complex thinking and multi-jump reasoning either. CoT and CR are set with one shot to balance the influence of the same setup in RFF-G. CoT generates a continuous chain of thoughts until the model answers the question while CR generates a few hints first, then generates simple questions and answers until the model thinks it's enough to answer the question. We also conducted two extra baseline not showing in the Game of 24: Least-to-Most (Zhou et al., 2022), which generates the subproblem pipeline first and then follow the pipeline to solve the problem and Give-me-Hint (Agrawal et al., 2024), which generates helpful hints to help LLM solve the problem. Both baselines are recent and typical to serve as convincing control groups.

Results

Table 2 shows the accuracy of Llama3-8B-instruct and Qwen-2.5-7B-Instruct on four datasets and our method RFF shows great advantages over the other methods. Meanwhile, there is another an interesting phenomenon: since the methods present great improvement of accuracy to CoT on GSM8K, ASDiv and MATH datasets, contributing to the better focus on details and relations about progressive prompting methods. However, CR fails to reach the average level of other methods on simple task SVAMP with 71.2% compared to 85.1%, and demonstrates better performance on hard task MATH with 40.2% to 36.2%. We carefully check the output of the CR and our RFF to and find that the detailed hints and question-answer pairs can be helpful in hard problems like MATH, but they lead to a harmful phenomenon of overthinking when facing simple problems like SVAMP. And our method RFF, benefiting from the State Checker, can avoid overthinking when the State Checker thinks the reasoning should be stopped. We also notice the gap between RFF and CoT is increasing with the base ability of the model decreasing (from Qwen to Llama), demonstrating a significant complementary effect on the model's reasoning ability.

4.3 Commonsense Problem Benchmark

To further explore the effectiveness of RFF in different NLP tasks, we have conducted experiments on commonsense problem benchmarks. Commonsense problems usually refer to those that require basic knowledge and reasoning ability in human daily life to solve. These problems rely on background knowledge, which is usually not explicitly stated in the problems. we have conducted experiments on two widely used commonsense benchmarks: CommonQA (Talmor et al., 2018) and LogiQA (Liu et al., 2020) using RFF-G. Both two benchmarks are multiple-choice, which contains 12102 and 8678 questions respectively, each with one correct answer and four choices.

Task Setup

We conduct this task on Llama3-8B-Instruct to test on this benchmark. Considering the nature of the commonsense tasks, we choose RFF-G to solve the question. We judge the accuracy of the answer by observing whether the model outputs the correct option, any other forms of answers will be viewed as wrong.

Baselines

For commonsense tasks, it will be much helpful if the model is given background informations before answering the question, so the hints based and progressive prompt reasoning paradigms will serve as effective references. Same as math benchmark, we choose COT, CR, Least-to-Most and CR as our experiment baselines.

Results

As shown in Table 3, all the reasoning paradigms achieve a better accuracy on CommonQA, demonstrating the complex prompting can easily improve the performance on it. However, the differences among these methods are not so apparent (all distributed at around 76%), so the sole result of CommonQA is not sufficient to express the superiority of our method. In LogiQA, the differences shows a greater gap among these baselines: the results of Least-to-Most and Give-me-Hint are close to COT while the results of RFF and CR shows a significant improvement over COT.

Method	CommonQA	LogiQA	AVG
COT	73.1%	41.8%	57.5%
CR	75.4%	**45.5%**	60.5%
Least-to-Most	74.3%	42.9%	58.6%
Give-me-Hint	76.6%	42.0%	59.3%
RFF	**77.1%**	45.2%	**61.2%**

Table 3: The results of commonsense reasoning benchmarks. The best result in each column is highlighted in **bold**, and the second best in each column is added an underline.

The results also indicate that in commonsen tasks, especially in easy task (CommonQA), simple prompting method (Give-me-Hint) achieves better scores than complex prompting methods (Least-to-Most and CR), which we think it's for the over thinking of these methods. Meanwhile, RFF can still maintain a high score due to its step-evaluation mechanism: when facing simple questions, the forward reasoning can quickly meets the backward reasoning, then the RFF turns to simple COT to solve the problems instead of over thinking.

4.4 Studies of Redundant Thinking

We investigate the limitations of traditional algorithms for solving Game of 24. While conventional breadth-first searching methods perform well in low-dimensional solution spaces, their unguided exploration mechanisms may lead to significant computational resource waste and efficiency degradation when handling higher-dimensional problems.

To validate this theoretical hypothesis, we constructed an experimental dataset comprising 100 enhanced problems (IDs 901-1000) by strategically adding the constant "1" to original four-number combinations, creating five-number variants. Theoretically, this operation preserves the solvability of problems (based on arithmetic identity transformations) and is expected to decrease the difficulty by introducing a redundant variant.

Model	Method	ACC	Visited States
GPT-4	CR(n=5)	76%	7.06
	RFF(n=5)	89%	**5.96**
	RFF(n=10)	**93**%	9.13
Llama3-8B	CR(n=5)	26%	96.56
	RFF(n=5)	85%	**28.62**
	RFF(n=10)	**92**%	56.13

Table 4: The results of 5 numbers of the Game of 24.

As shown in Table 4, after adding a redundant "1", the performance of CR decreased significantly with GPT-4 (from 94% to 76%). In contrast, RFF achieves a higher success rate with fewer visit states compared with CR. When we further expand the search space, the model's performance continues to improve. At the same time, we observe the smaller model requires more attempts to achieve performance comparable to the original data. However, RFF consistently surpasses CR in terms of success rate and resource consumption. The result

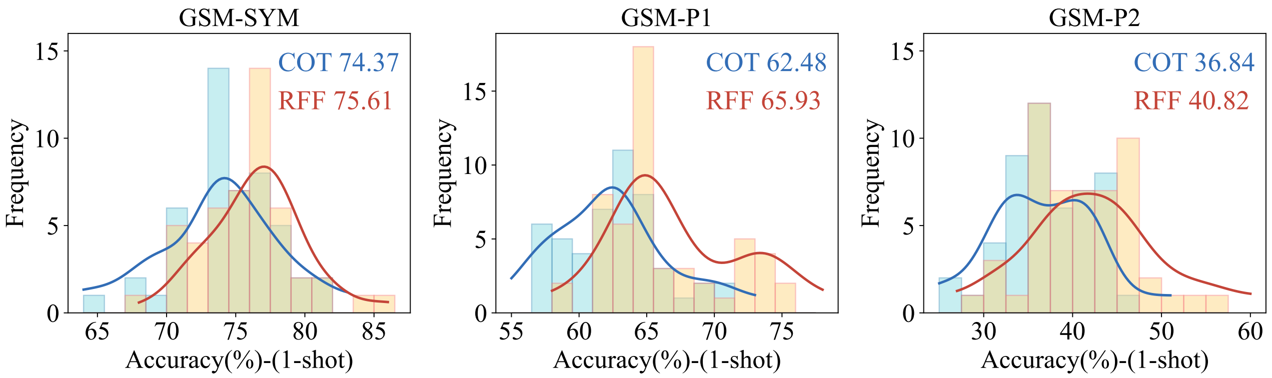


Figure 5: The result of CoT and RFF on GSM-Symbolic dataset. The score in the upper right of the chart is the average accuracy score of the 50 subsets. RFF shows more stable and better distributed at high accuracy.

demonstrates the effective prospective state space pruning in RFF and validates the superiority of the searching space convergence mechanisms of RFF.

4.5 Studies of Robust Thinking

To address the data leakage risks associated with the widespread adoption of the GSM8K, researchers (Mirzadeh et al., 2024) proposed the GSM-Symbolic benchmark through semantic-preserving transformations. This dataset generates derivative problems via entity/quantity substitution (GSM-SYM) and the addition of single (GSM-P1) or dual (GSM-P2) conditional constraints to the original question. Despite theoretical expectations that surface-level modifications (e.g., name/quantity changes) should not impact reasoning capabilities, empirical observations reveal significant accuracy degradation across all models. Following the standard evaluation protocol of GSM8K, we systematically assess the reasoning generalization of RFF models using the publicly available SYM, P1, and P2 subsets (each subset contains 50 variants from an original dataset). We employ Llama3-8B-Instruct with CoT and RFF methods to conduct this task.

Figure 5 shows the accuracy distribution of the 50 variants datasets in three subsets. The result exhibits that the accuracy of both methods drops on these three datasets, representing the fragility of the reasoning ability of models.

However, RFF still has advantages in the average accuracy and shows a more concentrated and more accurate distribution. The result emphasizes that the form of forward reasoning guided by backward reasoning is quite robust in the face of variant problems.

5 Conclusion

In this paper, we introduce Reason from Future (RFF), a novel reasoning paradigm aiming at enhancing the reasoning ability of LLMs for complex problems. RFF leverages a bidirectional reasoning framework integrates top-down planning with bottom-up reasoning accumulation to generate a solution path. This aids in the convergence of the search space for the model, thereby enhancing inference efficiency. Simultaneously, it allows the model to focus on critical information, which improves the accuracy of reasoning. RFF has demonstrated superior performance across both searching tree tasks (Game of 24) and directed acyclic graph tasks (math problems and commonsense problems), showing the potential to enhance the model's reasoning capabilities.

Limitations

The effectiveness of RFF relies on the model's ability for reverse thinking. Since the model has not been trained with specialized data, there can be rare instances where errors in the final step of reverse reasoning lead to failure. In future work, we will introduce fine-tuning or reinforcement learning to further enhance the generalizability of this reasoning paradigm.

Acknowledgments

This research was partially supported by National Natural Science Foundation of China under Grant No.12326612, No.82202984, Zhejiang Key R&D Program of China under Grant No.2024SSYS0026, Zhejiang Key Laboratory of Medical Imaging Artificial Intelligence, and the Transvascular Implantation Devices Research Institute (TIDRI).

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*.

Vansh Agrawal, Pratham Singla, Amitoj Singh Miglani, Shivank Garg, and Ayush Mangal. 2024. Give me a hint: Can llms take a hint to solve math problems? *arXiv preprint arXiv:2410.05915*.

Enric Boix-Adsera, Omid Saremi, Emmanuel Abbe, Samy Bengio, Etai Littwin, and Joshua Susskind. 2023. When can transformers reason with abstract symbols? *arXiv preprint arXiv:2310.09753*.

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*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Leonie Koban, Peter J Gianaros, Hedy Kober, and Tor D Wager. 2021. The self in context: brain systems linking mental and physical health. *Nature Reviews Neuroscience*, 22(5):309–322.

JDMCK Lee and K Toutanova. 2018. Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 3(8).

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint arXiv:2007.08124*.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2021. A diverse corpus for evaluating and developing english math word problem solvers. *arXiv preprint arXiv:2106.15772*.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2024. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*.

Alec Radford. 2018. Improving language understanding by generative pre-training.

Bilgehan Sel, Ahmad Al-Tawaha, Vanshaj Khattar, Ruoxi Jia, and Ming Jin. 2023. Algorithm of thoughts: Enhancing exploration of ideas in large language models. *arXiv preprint arXiv:2308.10379*.

Bilgehan Sel, Ruoxi Jia, and Ming Jin. 2025. Llms can plan only if we tell them. *arXiv preprint arXiv:2501.13545*.

R Nathan Spreng, Raymond A Mar, and Alice SN Kim. 2009. The common neural basis of autobiographical memory, prospection, navigation, theory of mind, and the default mode: a quantitative meta-analysis. *Journal of cognitive neuroscience*, 21(3):489–510.

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

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

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

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

Jundong Xu, Hao Fei, Liangming Pan, Qian Liu, Mong-Li Lee, and Wynne Hsu. 2024. Faithful logical reasoning via symbolic chain-of-thought. *arXiv preprint arXiv:2405.18357*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

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.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Yifan Zhang, Jingqin Yang, Yang Yuan, and Andrew Chi-Chih Yao. 2023. Cumulative reasoning with large language models. *arXiv preprint arXiv:2308.04371*.

Chuanyang Zheng, Zhengying Liu, Enze Xie, Zhenguo Li, and Yu Li. 2023. Progressive-hint prompting improves reasoning in large language models. *arXiv preprint arXiv:2304.09797*.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2023. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

A Detailed Difference between Baselines

Although most progressive prompting methods employ the reasoning paradigms by dividing a hard problem into several simple subproblems, the mechanism for decomposing the problem and the mechanism for how to solve the subproblems are different and serves as the core of the reasoning paradigms. We carefully compare the two typical baselines we used in our work (Least-to-Most and CR), and analyzed their characteristics and difference with RFF.

A.1 Difference between Least-to-Most

For Least-to-Most, it generates its sub-problems of the final problem in the very beginning, then the LLM follows the pipeline of the sub-problems to generate its reasoning. It's a two-stage plan-solve reasoning and the sub-problems never change during the solution, so the sub-problems may be wrong or insufficient because of a lack of the intermediate information generated by the reasoning process.

The biggest difference between RFF and Least-to-Most is that process of RFF starts from the end state of the problem, RFF doesn't plan just at the beginning but to continue to generate questions or hints or guidance by backward reasoning to lead LLM to better reason forward.

A.2 Difference between CR

For CR, it generates several hints for the problem first, the hints are helpful to solve the questions. Then the LLM continues to generate subproblem which based on the hints and then answer it until the LLM can solve the problem.

Differ from CR, which generates guidances from the previous visiting states to reduce hallucinations, our method (RFF) generates guidance from the backward reasoning to give the model a full-perspective of the problem. Additionally, the detailed hints and nonstop subproblems may lead CR to overthinking, meanwhile, the full-perspective and State Check ensure the ability of RFF to stop overthinking quickly when facing simple questions with complex prompting.

B Extra Experiments on Reasoning Paradigms

We compare two different backward reasoning strategies on Math Problems called Pair Reasoning (same as RFF in 2) and Single Reasoning as shown in Figure 6, we conduct experiments on GSM8K dataset using Llama3-8B-Instruct with greedy search.

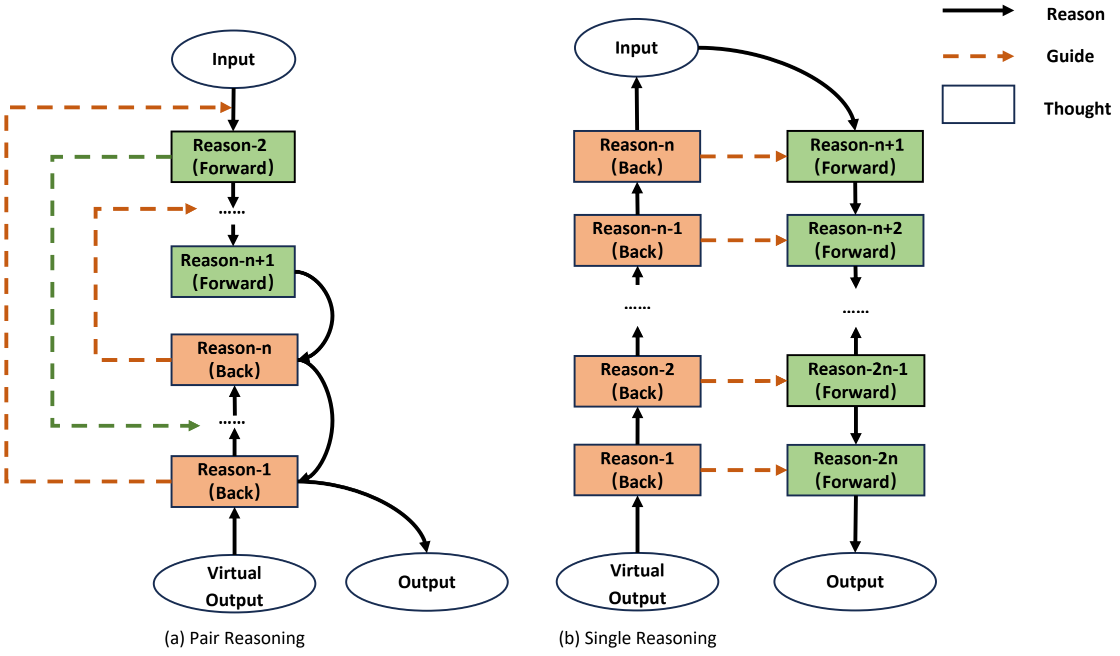
Model	Method	ACC
Llama3-8B	CoT	75.6%
	Pair Reasoning RFF	**83.8**%
	Single Reasoning RFF	69.8%

Table 5: The results of different back reasoning strategies.

As shown in Table 5, the performance of Single Reasoning RFF drops badly on GSM8K dataset and even worse than CoT, we assume that the weakness of the strategy of Single Reasoning is that when deducing the whole chain of backward thought, the situations of multi-hop seriously affects the backward reasoning without new information generated by the forward reasoning. So when employing backward reasoning paradigms, a continuous generation of new information is needed.

C Appendix for Prompts

The design of prompts is critical to lead the model to reason exactly according to the paradigm we have planned. We design these prompts deliberately to ensure the model reason and output according to the format we give it.



(a) Pair Reasoning

(b) Single Reasoning

Figure 6: Two different strategies of backward reasoning

System: Suppose you are one of the greatest AI scientists, logicians, and mathematicians. Let's play a game.
The Input is the current state, which contains four or three numbers.
The Target is the state we want to get, which contains one or two numbers.
The Last Step is how to get the Target with Input in the last step.
The Avoid is tested to be a wrong Last step, you need to generate another different step.
What you need to do is to think how to use Input to get Target use some steps, just output the most likely Last Step you think.
Notice:
1 Now do not calculate the game, you need to rely on your instincts to give the most likely Last Step directly, and do not output other thinking process.
2 The Last Step should contains two parts: "calculation" and "left".
3 The number used in "calculation" may not appear in Input, and the result of "calculation" must appear in "left".
4 The numbers in "left" must be the same as Target.
5 You are forbidden to output Steps, you should output Last Step only.

User:
Input: 1 3 6 11
Target: 24
Avoid: 3 x 8 = 24 (left: 24)
Avoid: 4 x 6 = 24 (left: 24)

Assistant:
Last Step: 2 x 12 = 24 (left: 24)

Figure 7: Prompts of Last Step Generator for Game of 24

System: Now you are given few examples about the game:
Input is the current state of the game.
Target is the final state you need try to satisfy using basic arithmetic operations (+ - * /) with the Input.
Steps are how to get Target with Input through basic operations.
Next Step is the how to get the Target with Input in the next step.
The Avoid is tested to be a wrong Next step, you need to generate another different step.
You need to choose two numbers from Input and use one basic arithmetic operations (+ - * /) to generate a new number.
Notice:
1 Output the Next Step directly and do not output the other thinking process.
2 The Next Step contains and only contain two parts: "calculation" and "left".
3 The "left" should be close to Target but not asked to be the totally same.
4 Your calculation must be correct.
5 Do not output Steps.

User:
Input: 8 8 10 12
Target: 8 16
Avoid: 8 + 8 = 16

System:
Next Step: 12 - 10 = 2 (left: 2 8 8)

Figure 8: Prompts of Stepwise Forward Reason for Game of 24

System: You are one of the GREATEST mathematicians, logicians, programmers, and AI scientists. You are intelligent and rational. You are prudent and cautious. You THINK NATURAL, BROAD AND DEEP. Let's think step by step.
You will be given a mathematical problem, a question about it and the information we have calculated.
Notice:
1 You are not permitted to solve the question from the beginning.
2 You need to analyse the question and figure out what will be the last step to solve the question.
3 Make sure your analysis are used to calculate the result of the question not the intermediate result.
4 Output the calculation process, and the information we need.

User:
Problem: {problem}
Question: {question}
Information: {information}

Assistant:
To know {question}: Since {information}, we can do: {calculation process}
Need Information: {need information}

Figure 9: Prompts of Last Step Generator for math problems

System: You are one of the GREATEST mathematicians, logicians, programmers, and AI scientists. You are intelligent and rational. You are prudent and cautious. You THINK NATURAL, BROAD AND DEEP. Let's think step by step.
You will be given a mathematical problem, a question about the problem, and the information we have calculated.
Notice:
1 You are not permitted to solve the question directly.
2 You need to analyse the question and figure out next step we get to solve the question.
3 Make sure your analysis are used to calculate the result of the question not the intermediate result.
4 Output the calculation process, and the new information we get.

User:
Problem: {problem}
Question: {question}
Information: {information}

Assistant:
Next Step: {calculation process}
New Information: {new information}

Figure 10: Prompts of Stepwise Forward Reason for math problems

System: You are one of the GREATEST mathematicians, logicians, programmers, and AI scientists. You are intelligent and rational. You are prudent and cautious. You THINK NATURAL, BROAD AND DEEP. Let's think step by step.
You will be given a mathematical problem, a question about it and information we have calculated.
Notice:
1 You are not permitted to solve the question.
2 You need to analyse the question and information, then figure out whether we have already solved the question.
3 Make sure your analysis do not consist of calculation process.
4 If we have solved the question, you should output [True], else you should output [False].

User:
Problem: {problem}
Question: {question}
Information: {information}

Assistant:
Analyse: {analyse}
Answer: [True, False]

Figure 11: Prompts of State Check for math problems