
Toward Adaptive Reasoning in Large Language Models with Thought Rollback

Sijia Chen¹ Baochun Li¹

Abstract

Large language models (LLMs) have been routinely used to solve various tasks using step-by-step reasoning. However, the structure of intermediate reasoning steps, or *thoughts*, is rigid and unidirectional, such as chains, trees, or acyclic-directed graphs. Consequently, the resulting inflexible and forward-only reasoning may not address challenging tasks and fail when the LLM frequently gives false responses, i.e., “hallucinations”. This paper proposes a new reasoning framework, called *Thought Rollback* (TR), allowing LLMs to adaptively build thought structure while maintaining effective reasoning toward problem-solving under “hallucinations”. The core mechanism of TR is *rolling back thoughts*, which allows LLMs to perform error analysis on thoughts, and thus roll back to any previously mistaken thought for revision. Subsequently, by including such trial-and-error in the prompt to guide the LLM, each rollback leads to one more reliable reasoning path. Therefore, starting with a simple prompt without human annotations, LLM with TR adaptively and gradually explores thoughts for a correct solution. Comprehensive experiments on mathematical problems and multi-task reasoning demonstrate the state-of-the-art performance of TR in terms of problem-solving rate and interaction cost. For instance, the solving rate of GPT-4 with TR outperforms the current best by 9% on the MATH dataset. The source code is available under the folder *examples/ThoughtRollback* of <https://github.com/iQua/llmpebase>.

¹Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ontario, Canada. Correspondence to: Sijia Chen <sjia.chen@mail.utoronto.ca>.

1. Introduction

Large Language Models, initially designed for text generation with autoregression, are widely recognized to excel in a diverse array of natural language processing (NLP) tasks. Yet, at a particular model scale, their reasoning abilities, particularly in scaled-up versions like GPT-4 (OpenAI, 2023) and Llama 2 (Touvron et al., 2023), heavily depend on prompt engineering. With well-crafted prompts — even just a simple *Let’s think step by step* (Kojima et al., 2022) — LLMs are able to perform step-by-step reasoning and achieved noteworthy success in mathematical, symbolic, and common sense tasks. With reasoning, LLMs are capable of producing coherent language sequences, called *thoughts*, which serve as intermediate reasoning steps toward solving the problem at hand. Extended from simple chain reasoning (Wei et al., 2022) with linear left-to-right thoughts, more complex reasoning became feasible in recent works by establishing thought structures that resembled trees (Yao et al., 2023) and graphs (Besta et al., 2023; Zhang et al., 2023; Luo et al., 2024).

However, existing thought structures are unidirectional and thus allow a forward-only reasoning process, meaning that thoughts are generated sequentially from the start to the end. The efficacy of this reasoning process hinges on a redundant and, consequently, inefficient thought structure, requiring thorough explorations of each thought before progressing to the next. One major drawback of forward-only reasoning is that errors can propagate quickly (Yu et al., 2024). Consider the common case where one thought is incorrect or inaccurate: with forward-only reasoning, all thoughts derived from it can be misled. Even with revisions based on step-by-step evaluations (Weng et al., 2023), such propagation of errors can introduce further deviations from the correct path of reasoning, since LLMs have been found to confidently provide false information (i.e., “hallucinations” or “laziness”) (Jiang et al., 2023).

Indeed, humans also provide false information as frequently and randomly as LLMs do during reasoning but can still solve challenging problems eventually. This is attributed to adaptive reasoning, in which one does not pre-defined a fixed structure for thoughts and does not simply deduce forward but adaptively adjusts the thought structure after evaluating errors during reasoning. Such reasoning enables

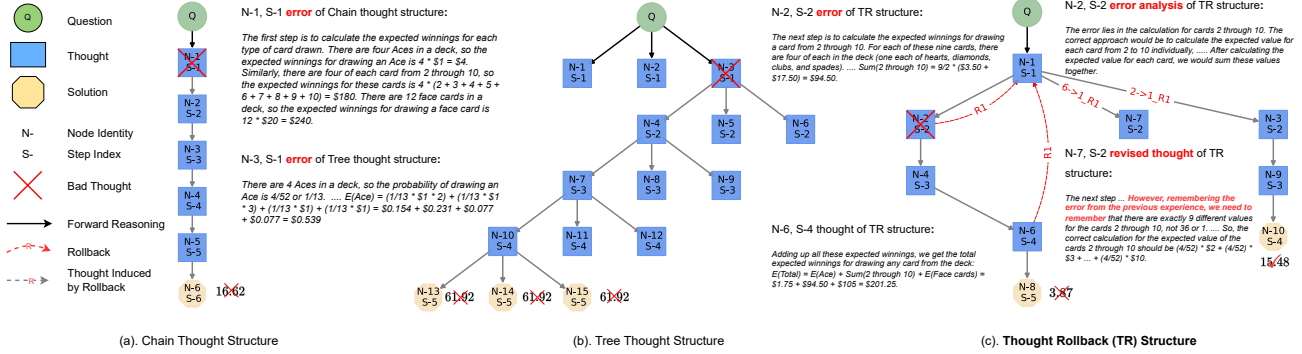


Figure 1. Schematic illustrating thought structures for problem solving with GPT-4. The chain, tree, and our thought rollback (TR) structures are plotted based on the NetworkX lib (Hagberg et al., 2008). The question from the MATH dataset (Hendrycks et al., 2021b) is: I draw a card from a standard 52-card deck. If I draw an Ace, I win 1 dollar. If I draw a 2 through 10, I win a number of dollars equal to the value of the card. If I draw a face card (Jack, Queen, or King), I win 20 dollars. If I draw a ‘clubsuit’, my winnings are doubled, and if I draw a ‘spadesuit’, my winnings are tripled. ... What would be a fair price to pay to play the game? In (c), we present a partial thought structure built by GPT-4 with TR and place the full version of the TR structure in Figure 4 of the Appendix.

humans to begin with one simple or wrong thought but frequently introspect during reasoning, that is, to reconsider previous steps and build new reasoning paths from these reflections. In this paper, we argue that this iterative error correction nature of adaptive reasoning is essentially supported by **rollback** — jumping to previous steps with a new experience to reconsider the reasoning.

Therefore, we propose a new reasoning framework, Thought Rollback (TR), relying upon the rolling back of thoughts to enable the adaptive reasoning of LLMs for general problem solving. TR embraces a *rollback controller* and a *prompt enhancer* that works seamlessly to enable the LLMs to generate an effective thought structure from one thought derived from a simple input prompt, as shown by Figure 1.

LLMs with TR start with generating one thought from a simple zero-shot prompt containing only the question description. Subsequently, for each generated thought, the *rollback controller* allows the LLM to analyze the obtained chain of thoughts and thus determines whether to roll back and to which previous thought. Once rollback is triggered, *prompt enhancer* accumulates this error analysis as experience in the prompt. As a result, by avoiding making similar mistakes mentioned by experience, LLM is able to generate a new and more effective reasoning path from the chosen thought. Therefore, “hallucinations” that occur in thought or analysis of LLM may not influence reasoning due to the continuous thought revision guaranteed by the iterative rollbacks during reasoning. For example, in Figure 1, different from chain (Wei et al., 2022) and tree (Yao et al., 2023) structures, which assume a fixed and unidirectional structure, reasoning with rollbacks enables LLM to build a thought structure adaptively and revise thoughts to achieve complex but reliable reasoning. Specifically, after reaching the $N-2 S-2$ and $N-6 S-4$, LLM finds an error in 2-th step

$N-2 S-2$ and thus rolls back to 1-th step $N-1 S-1$ to create two new reasoning paths. The rollback $N-6 \rightarrow N-1$ leads to the revised thought $N-7 S-2$. The rollback $N-2 \rightarrow N-1$ utilizes the error analysis to enhance the prompt and obtains $N-3 S-2$, leading to a correct answer 15.48 compared to the previous mistaken 3.87.

We observe four contributions of TR. First, it is a lightweight and generalized automatic prompting framework. TR allows LLMs to perform complex reasoning effectively on various tasks without introducing task-specific human annotations in the prompt or additional human-made designs. Second, the performance of TR is robust to the “hallucinations” as LLMs are able to reconsider and revise any existing thoughts adaptively and repeatedly during reasoning. Thus, third, TR is cost-friendly as the thought structure is built progressively to reach a solution instead of relying on bulky search structures (Yao et al., 2023) or question analogies (Yu et al., 2024). Finally, our evaluation of TR on mathematical problems and multi-task reasoning demonstrates that TR outperforms some state-of-the-art approaches while maintaining a lower interaction cost.

2. Related Work

By only guiding the reasoning behavior of LLMs, such as GPT-4 (OpenAI, 2023) and Llama2 (Touvron et al., 2023) with the text prompt, prompt engineering (Brown et al., 2020; Kojima et al., 2022) is parameter efficient and often matches or surpasses fine-tuning performance. Therefore, plenty of work has been proposed to enable LLMs to perform **multi-step reasoning** containing a series of intermediate steps, each known as the thought presented as a text sequence. Starting from chain-of-thought (CoT) (Wei et al., 2022) prompting, which provides reasoning examples in the prompt to deduce a chain of thoughts, subsequent endeavors,

especially SC (Wang et al., 2022) and Complex CoT (Fu et al., 2023) augment the chain reasoning. Recent advances extend the chain structure into structured thoughts. ToT (Yao et al., 2023) and BoT (Sijia et al., 2024) pre-defines the thought structure as a tree, thus supporting exploring multiple thoughts in each step before generating the next. The graph of thoughts (GoT) (Besta et al., 2023) and cumulative reasoning (CR) (Zhang et al., 2023) further instantiate thoughts toward a solution as the graph structure. Another line of work focuses on the thought structure that allows the thought ensemble in each step. Thought propagation (Yu et al., 2024) explores analogous problems and then aggregates their results to update the solution for the given question, leading to a radial thought structure. To the best of our knowledge, none of the existing work supports the cyclic structure to allow LLMs to revise previous thoughts or recreate a new reasoning path from the previous step after being blocked at the current reasoning step. We fill this gap by proposing the rollback of thought, leading to a thought structure of directed cyclic graphs.

Despite these achievements, LLMs often struggle with complex tasks, primarily due to the frequent occurrence of “hallucinations”—producing false outputs (Jiang et al., 2023; Yu et al., 2024), and “laziness”—yielding invalid or no output. Therefore, after noticing that LLMs have **self-verification** (Weng et al., 2023; Madaan et al., 2023) abilities and thus can analyze the answer for further correcting the errors in reasoning (Zheng et al., 2023; Wu et al., 2024). However, the most recent work (Huang et al., 2024) argues that LLMs cannot self-correct their reasoning, emphasizing the invalidity of applying simple verification to the reasoning path. Thus, most recent work either builds iterative-based verification (BoT (Sijia et al., 2024)) or focuses on **step-by-step verification** (Ling et al., 2023; Lightman et al., 2024). Combining these insights, we aim to exploit LLMs to analyze intermediate thoughts during reasoning to correct these thoughts and adjust reasoning direction adaptively. Continuous verification and revision may eliminate the negative impact of “hallucinations” or mistakes on the solutions.

Another related research stream is **automatic prompting** (Kojima et al., 2022; Zhang et al., 2022), which automatically constructs effective prompts to facilitate reasoning without human-made and task-specific demonstrations. As LLMs can learn from mistakes to become better reasoners (An et al., 2023; Sijia et al., 2024), this paper also releases human efforts from the prompt design by boosting the prompt with the error analysis of thoughts. We also show that by accumulating error analysis in the prompt during reasoning, LLMs are able to avoid making similar mistakes and explore correct solutions with interaction cost far less than ToT (Yao et al., 2023) and BoT (Sijia et al., 2024).

3. Preliminary

3.1. Problem Statement

Given a pre-trained large language model (LLM) denoted as $f(\cdot)$, the prompt engineering is to design the prompt $\mathbb{I}(\cdot)$ to make the model perform desired reasoning behavior toward addressing the given problem x . Specifically, multi-step reasoning contains T intermediate steps $z_{0..T} = [z_0, z_1, \dots, z_T]$ to bridge $z_0 := x$ and the answer $z_T := y$. To get $z_{0..T}$, we focus on step-wise thought generation in which each thought is a coherent language sequence z_n , behaving as one intermediate reasoning step, and z_n is generated as $z_n \sim f(z_n | \mathbb{I}(z_{0,1..n-1}))$. Therefore, as thought is the LLM’s output, we can define the bad thought caused by the “hallucinations”, “laziness”, or false reasoning of LLMs as \hat{z}_n .

These generated thoughts $z_{0..T}$ naturally follow a specific structure, such as a chain or tree. These structures are unidirectional and thus only support **forward-only reasoning**, which proceeds in a linear, sequential manner, meaning that LLMs only generate the subsequent thought z_{n+1} from z_n . For instance, any edge $e_{n,m}$ of the structure is limited to $m = \{n, n + 1\}$ while $m > n$.

This paper focuses on alleviating the effect of bad thoughts on the solution by making LLM not simply perform forward-only reasoning but achieve *adaptive reasoning*, which allows LLMs to 1) start from a simple prompt $\mathbb{I}(z_0)$, 2) self-organize the thought structure adaptively during reasoning, and thus 3) when \hat{z}_n occurs, LLM can make revisions and create better new reasoning paths till getting the solution. Specifically, not only advancing the reasoning sequentially, any previously generated thought will be reconsidered by continuously rolling back from n -th thought to one previous m -th thought, where $m \in [0, n - 1]$.

3.2. Motivation: Forward-only reasoning fails in bad thoughts

Forward-only reasoning may fail as the bad thought \hat{z}_n is caused by the following three cases of **error propagation**.

Case $[\hat{z}_m, \hat{z}_{m+1}, \dots, \hat{z}_n]$. A bad or illogical thought \hat{z}_m leads to all subsequent errors, where $\hat{z}_n \sim f(\hat{z}_n | \mathbb{I}(\hat{z}_{0,\dots,m..n-1}))$ and $m < n$.

Case $[\hat{z}_m, z_{m+1}, \dots, \hat{z}_n]$. \hat{z}_m does not lead to direct mistakes but causes a bad thought \hat{z}_n after many steps. For instance, this appears when \hat{z}_m behaves as one part of a solution.

Case $[z_{m..n-1}, \hat{z}_n]$. A bad thought \hat{z}_n may arise from one previous correct thought z_m because the wrong reasoning direction appears from this step.

The chain reasoning, typically in Chain-of-thought (CoT) prompting, generates a chain of thoughts $z_{0,1..T}$ (Wei et al.,

4.2. Rolling Back Thought with Reasoning Analysis

During reasoning, *rollback controller* enables an adaptive rollback by exploiting LLM to determine the rollback $n \rightarrow m$. However, making LLM know the concept of rollback may introduce unnecessary complexity. Thus, TR supports the rollback mechanism by performing error analysis on thoughts. Specifically, with a task-agnostic prompt $\mathbb{I}_R(R, [z_{0\dots n}])$, where R is a common error analysis instruction, LLM is guided to analyze a thought chain $[z_{0\dots n}]$, leading to the error analysis $A_m^n \sim f(A|\mathbb{I}_R(R, [z_{0\dots n}]))$. Eventually, LLM is able to identify the indexes \widehat{M} of bad thoughts $\widehat{z}_{m \in \widehat{M}}$.

To get which thought to roll back to from z_n , TR follows the rule to roll back to the one step before the first bad thought. There are two reasons for this. First, generating the next thoughts from a bad thought is unreasonable. Second, we aim not to remove the bad thought but to generate a new reasoning path. Thus, we choose z_{m-1} as the rollback destination, where $m = \arg \min \widehat{M}$. Besides, one thought will not be selected as the rollback destination more than U times to avoid all subsequent thoughts rolling back to the same previous thought. Thus, when the number of rollbacks to a thought reaches U , the next earliest one of $m = \arg \min \widehat{M} \setminus \left\{ \arg \min \widehat{M} \right\}$ will be selected.

Therefore, when \widehat{M} is not empty, TR generates the next thought z_m^n from $m - 1$, meaning that a new reasoning path $[z_{0\dots z_{m-1}}, z_m^n]$ derived from the rollback $n \rightarrow m$ is created for the thought structure. As the existing $z_{0\dots n}$ remains unchanged, the reasoning continues by generating $z_{0\dots n+1}$. For ease of description, we define $n \rightarrow m$ as the *outgoing rollback* for $z_{0\dots n+1}$ and the *incoming rollback* for the new reasoning path $[z_{0\dots z_{m-1}}, z_m^n]$.

4.3. Enhancing the Prompt with Errors as Experience

Through iterative rollback from the n -th to the $m - 1$ -th step, TR gains the opportunity to address the three scenarios outlined in subsection 3.2. However, as discussed in 3.3, regenerating a next thought z_m^n based on the same prompt may repeat existing mistakes in the new reasoning path. Especially considering that TR is built upon the prompt containing no human annotations, the thought regeneration after the rollback is equivalent to randomly exploring z_m^n as in unidirectional structures.

Therefore, TR embraces *prompt enhancer* to also roll back the error analysis A_m^n to the $m - 1$ -th thought. Unlike BoT (Sijia et al., 2024) with outcome analysis, which utilizes error feedback on the final result, TR performs process analysis, i.e., rollback-by-rollback verification, to get error reports on intermediate thoughts, guiding the subsequent thought generation. With error analysis, each rollback is

regarded as a trial on generating subsequent thoughts for z_{m-1} because the analysis contains a trial experience: *what mistakes may appear in the following steps of z_{m-1}* . By including A_m^n as an experience in the prompt, LLMs can avoid making existing bad thoughts after learning from mistakes. Eventually, each rollback $n \rightarrow m$ creates the error experience A_m^n .

Experience accumulation. The thoughts $\mathbf{Z}_{z_{0\dots q-1}}^{\chi(\cdot)} = \{z_j^i | z_j^i \in z_{0\dots q-1}, z_i \notin z_{0\dots q-1}, j \in [0, q-1], i \in \chi(j)\}$ of a reasoning path $z_{0\dots q-1}$ may derive from multiple *incoming rollbacks*, where $\chi(j)$ is the set of rollbacks whose destination is j -th thought of this path. As each rollback creates an error experience from one trial of the given question, *incoming rollbacks* lead to a series of experiences $A_{z_{0\dots q-1}}^{\chi(\cdot)}$. By accumulating an ensemble of trial-and-error reasoning experiences as the in-context learning examples in the prompt, LLM will learn from more experiences to generate the correct next thought $z_q \sim f\left(z_q | \mathbb{I}\left(A_{z_{0\dots q-1}}^{\chi(\cdot)}, z_{0\dots q-1}\right)\right)$, as shown in Figure 2 and two examples of Figure 8 and Figure 10.

4.4. Ensembling Solutions

TR may create massive final solutions as each adaptive triggered rollback leads to one more new reasoning path toward answering the question. Thus, we directly stop reasoning when there is K number of solutions $\{z_{0\dots T_k}\}_{k=1}^K$ obtained. Then, weighted majority voting (W-Voting) will be performed on them for a final solution. Specifically, for the solution z_{T_k} , the weight w_t is higher when 1) it has a lower number of *outgoing rollbacks* denoted as α_{T_k} , meaning that fewer bad thoughts are identified; and 2) more experiences $\beta_{T_k} = |A_{z_{0\dots T_k}}^{\chi(\cdot)}|$ are accumulated along this reasoning path. Eventually, TR outputs the final solution as: $\arg \max_{v \in V} \sum_{k=1}^K I(v_k = v) (\beta_{T_k} - \alpha_{T_k})$, where V is the collection of solutions and v_k is the value of k -th solution.

5. Experiments

Datasets. We conduct experiments on two streams of tasks. For the mathematical problems, we evaluate the performance of TR on test sets of GSM8K₁₃₁₉ (Cobbe et al., 2021), SVAMP₃₀₀ (Patel et al., 2021), AQUA-RAT₂₅₄ (Ling et al., 2017), MATH₉₀₀ (Hendrycks et al., 2021b), TheoremQA₄₀₀ (Chen et al., 2023b) datasets, where numerical subscripts indicate sample size. For TheoremQA₄₀₀, we specifically use half of the test set without visual information, leading to 400 samples. Following ToT (Yao et al., 2023), we utilize 100 challenging games of Game of 24. For multi-task reasoning, such as symbolic reasoning, we extract 900 samples from 56 categories of MMLU (Hendrycks et al., 2021a), i.e. MMLU₉₀₀.

Table 1. Evaluating the reasoning ability of TR with GPT models under four well-known mathematical problems. TR is specifically evaluated against leading methods such as Faithful-CoT (Lyu et al., 2023), and CSV (Zhou et al., 2024), each achieving state-of-the-art (SOTA) performance on the SVAMP and MATH datasets, respectively. The best results, apart from the SOTA, are in bold. The 5 shots of CoT examples used by our TR experiments are extracted randomly from the trainset of the same category.

Methods	ZeroShot	GPT-4				GPT-3.5-turbo			
		GSM8K	SVAMP	AQuA-RAT	MATH	GSM8K	SVAMP	AQuA-RAT	MATH
SOTA	✗	98.7 ₈	95.4	85.04 ₈	84.32	89.2	84.3	60.6 ₈	40.56
ZeroShot	✓	87.1	79.33 [†]	50.4	42.2	76.3	74.8	53.5	24.5 [†]
ZeroShot-CoT	✓	93.1 [†]	84.67 [†]	73.2	44.7	79.6	77.5	53.9	30 [†]
CoT	✗	94.2 ₅ [†]	91.9 ₅	75.2 ₈	48.9 ₈	87.4 _{5,sc15}	83 ₅	59.4 ₅	-
C-CoT	✗	94.9 ₈	90.5 ₈	77.5 ₈	50.4 ₈	82.8 ₈	81.0 ₈	57.4 ₈	34.1 ₈
PHP+C-CoT	✗	95.5 ₈	91.9 ₈	79.9 ₈	53.9 ₈	85.1 ₈	83.1 ₈	60.6 ₈	36.5 ₈
BoT	✓	97.1	92.67	81.5	62.44	-	-	-	40.56
BoT+CoT	✗	98.7 ₈	95 ₈	85.04 ₈	66.33 ₈	-	-	-	-
Chain Reasoning [†]	✓	89.76	80.33	74.41	45.44	76.72	71.67	47.64	26.89
ToT Reasoning [†]	✓	90.9	84	76.38	48	79.83	78.33	54.72	30.44
TR [†]	✓	94.24	89	79.92	55	82.49	77.67	56.69	32.78
TR + CoT ₅ [†]	✗	96.06	91.33	84.25	62.56	86.5	79.67	57.87	31.44
TR + W-Voting [†]	✓	96.36	93	87.8	71.89	85.9	82.33	63.39	39.78
TR + CoT ₅ + W-Voting [†]	✗	96.97	93.33	87.4	72.11	87.79	82.67	62.6	35.89

Large language models. We utilize GPT-3.5-turbo (gpt-3.5-turbo-16k-0613), GPT-4 (gpt-4-1106-preview) (OpenAI, 2023) and Llama2 (Touvron et al., 2023), including Llama2-13b (Llama-2-13b-chat-hf) and Llama2-70b (Llama-2-70b-chat-hf) where 1b means one billion parameters. For LLMs with TR, the default settings for temperature and top-p are 0.7.

Baselines. Apart from zero-shot prompting (Kojima et al., 2022), the comparison approaches include Chain-of-thought (CoT) (Wei et al., 2022), SC (Wang et al., 2022) and Complex CoT (Fu et al., 2023) (C-CoT), where the subscript 5 or 8 indicates the number of shots while the subscript *sc* denotes the number of sample paths. Also, TR is compared with the related approaches, such as Boosting of thoughts (BoT) (Sijia et al., 2024), Tree of thoughts (ToT) (Yao et al., 2023), Cumulative Reasoning (CR) (Zhang et al., 2023), and Progressive-Hint Prompting (PHP) (Zheng et al., 2023). ToT follows the best first search (BFS). The breadth limit of ToT is 6 while BoT performs 10 boosting iterations on 15 binary trees. We also include the state-of-the-art (SOTA) methods, such as CSV (Zhou et al., 2024) that relies on GPT-4 Code Interpreter, on each dataset as an additional comparison. We set $K = 8$ for possible early stopping of TR in all experiments.

Metrics. All experiments report the Solve Rate (%) of the questions. We make LLM explicitly report the solution value after the strings, such as “The solution is” and “The choice is” in the $z_{0..T}$. Thus, the value is directly extracted and compared with the ground truth. The Interaction Number refers to the frequency at which we must consult the LLM until we receive conclusive responses.

Reproducibility. The results and methods marked with a superscript [†] are the results we obtained based on the open source platform *llmpebase*. Others without such a tag are collected from existing work, as shown in the Appendix’s subsection C.1.

5.1. Main Evaluation Results

Adaptive reasoning. With zero-shot prompting and no pre-defined thought structures, such as chain of Chain Reasoning and the tree of ToT Reasoning, TR allows GPT-3.5-turbo, GPT-4, and Llama2 to self-organize and explore thought structures toward answering the question. Under challenging tasks, LLMs with TR adaptively build complex structures, as shown by examples in the Appendix, by continuously rolling back from thoughts with “hallucinations”. For simpler tasks, lightweight structures are built by LLMs with TR. As such, with the ability to adjust thoughts and prompt the LLMs with accumulated experience of errors during reasoning, TR achieves a high solving rate and relatively lower resource cost.

Overall comparison. We show, especially in Table 1, that compared to existing multi-step reasoning approaches, LLMs with TR achieve the best and the second best solving rate on AQuA-RAT & MATH and GSM8K & SVAMP, respectively. Meanwhile, contrary to the resource-cost SOTA ones, such as BoT, which undertakes reasoning through massive tree thought structures, and CSV, which relies on GPT-4 Code Interpreter, TR yields notable performance by interacting less with relatively simpler LLMs. First, TR surpasses BoT by 6.3 on AQuA-RAT and 9.45 on MATH using GPT-4. In particular, TR requires only around 40 in-

teractions compared to the 500+ interactions of BoT. Using TR with zero-shot prompting, GPT-4 and GPT-3.5-turbo outperform the ones using few-shot CoT prompting and self-consistency. Under hard math problems, especially MATH, the solving rate of TR is 17.99 and 3.28 higher than PHP+C-CoT under GPT-4 and GPT-3.5-turbo, respectively. Second, LLMs with TR adaptively explore thought structures, which is significantly better than pre-defined forward-only Chain Reasoning and ToT Reasoning. After large-scale interactions with LLMs, the performance of the latter two zero-shot prompting methods only approaches the 8 shots Complex CoT. Figure 3 (a) shows that compared to ToT reasoning, our TR requires one-third or less of the interactions to achieve a new state-of-the-art. Finally, with an average 28 interactions with GPT-4, TR yields a competitive solving rate 87.56 on the multi-task dataset MMLU, which contains symbolic reasoning.

We emphasize that TR is more effective in challenging problems, as shown in Table 2 and Table 3. In the level 5 difficulty of MATH, GPT-4 with TR is only 2.84 lower than the CSV that embraces the GPT-4 Code Interpreter as an auxiliary. The solving rate of GPT-4+TR is 4.35 higher than the current best in TheoremQA. Along with better performance, interaction cost is reduced to an acceptable range by TR. Another observation is that LLMs with TR introduce more interactions in hard problems than the simpler ones. For instance, as shown in Figure 3 (a) and the first two columns of Table 2, the average interaction cost increases to around 60. Also, in Table 3, the results of the Game of 24 dataset show that GPT-4 with TR requires an average of 32 interactions to reach a solving rate of 87%, which is only 7% lower than the CR₂ (Zhang et al., 2023). This is a remarkable achievement as CR-related approaches rely on human-made demonstrations while TR is zero-shot prompting. Moreover, introducing the CR’s demonstrations into the prompt of TR increases the solving rate to 93% while reducing the number of interactions to 24. In addition, the better performance of TR + CR-Prompt shows that including demonstrations reduces the reliance on majority voting.

Effect of the rollback of thoughts. In Figure 3 (b) and (c), we specifically present the relation between rollbacks and the solving rate of reasoning paths and the decrease in the failure rates at the first step of the Game of 24. We define a reasoning path $z_{0..T}$ as *In Rollback* if a majority of its thoughts, represented by $Z_{z_{0..T}}^{X(\cdot)}$ in the subsection 4.3, are derived from *incoming rollbacks*. $z_{0..T}$ is defined as *Out Rollback* if more than two of its thoughts trigger *outgoing rollbacks* and as *No Rollback* if it includes no rollbacks. Figure 3 (b) presents these three types of reasoning paths TR generates during reasoning in four datasets. As TR allows the error analysis of each rollback to be accumulated in the prompt, as discussed in subsection 4.3, a *In Rollback*

Table 2. Evaluating TR with the ZeroShot setting on challenging mathematical problems and multi-task reasoning. With GPT-4, the existing SOTA zero-shot methods on the level 5 difficulty of the MATH, TheoremQA, and MMLU are from CSV (Zhou et al., 2024), PoT (Chen et al., 2023a), and BoT (Sijia et al., 2024), respectively. We use 324 samples out of 1324 for the MATH dataset. The average interaction number with LLM to solve each problem is reported within () behind the number.

Methods	MATH-level5 ₃₂₄	TheoremQA ₄₀₀	MMLU ₉₀₀
SOTA	55 (3)	52.4 (1)	93.2 (900+)
Llama2-13b [†]	2.47 (1)	9.3	51.78 (1)
Llama2-70b [†]	8.64 (1)	25.5	65.44 (1)
GPT-4 + ZeroShot-CoT [†]	23.46 (1)	43.75 (1)	82.33 (1)
GPT-4 + Chain Reasoning [†]	22.53 (11)	36.5 (9)	78.11 (6)
GPT-4 + ToT Reasoning [†]	24.38 (150)	38.25 (110)	79.22 (70)
Llama2-70b + TR + W-Voting [†]	12.65 (36)	29 (34)	58.33 (22)
GPT-3.5-turbo + TR + W-Voting [†]	20.06 (38)	39.5 (30)	70.11 (18)
GPT-4 + TR [†]	31.48	46.25	84.67
GPT-4 + TR + W-Voting [†]	52.16 (62)	56.75 (56)	87.56 (28)

Table 3. Utilizing TR with GPT-4 achieves competitive performance while maintaining low interaction cost on Game of 24 dataset.

Method	Solving rate	#Interactions	Generate tokens	Prompt tokens
Standard	7.3	1	-	-
Standard (best of 100)	33	100	1.8k	1k
CoT	4	1	-	-
CoT (best of 100)	49	100	6.7k	2.2k
CoT-SC	9 _{acc:100}	100	-	-
ToT(b=5)	74	30	5.5k	1.4k
CR(b=2)	94	27.4	-	-
CR(b=5)	98	29.72	-	-
BoT	83.7	724	15.8k	18.6k
BoT+CoT ₅	84.9	543	11.2k	15.5k
Chain Reasoning [†]	5	3	0.14k	0.22k
ToT(b=5) Reasoning [†]	25.6	36	2.82k	1.63k
TR [†]	70	32	5.96k	9.98k
TR + W-Voting [†]	87	24	5.03k	8.1k
TR + CR-Prompt [†]	86	24	5.03k	8.1k
TR + W-Voting + CR-Prompt [†]	93	24	5.03k	8.1k

path generally benefits from exploiting more experiences during reasoning. Therefore, as Figure 3 (b) verifies, *In Rollback* paths consistently correspond to a higher solving rate because of prompting LLMs with these trial-and-error experiences. On the contrary, those *Out Rollback* paths have significantly low solving rates because they include more bad thoughts (“hallucinations”), which consequently trigger more rollbacks after being identified by LLMs. Similarly, when the first step of Game of 24 derives from the number of 0 to 5 *incoming rollbacks*, the failing rate decreases significantly from higher than 0.8 to lower than 0.3. The final observation is that longer spans of rolling back are more important for revising thoughts. Figure 3 (c) shows that the first step caused by a rollback 3 → 0 has a higher success rate than the one caused by 2 → 0 and 1 → 0. Therefore, we argue that **a rollback, especially generated from more latter reasoning steps (thoughts), contribute more to the thought revision.** This may be because the error analysis brought by rollbacks of later reasoning steps contains more information and is more helpful in improving prompts.

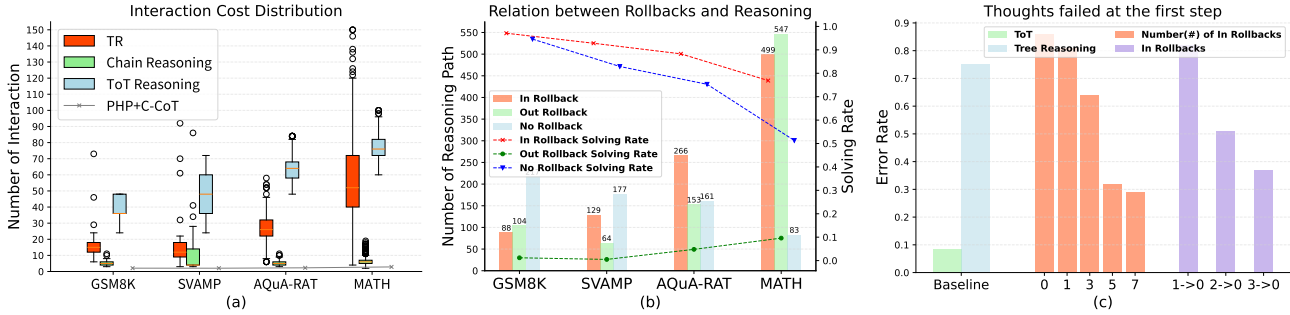


Figure 3. Effectiveness of TR on interaction cost saving and thought revision through the rolling back of thoughts. a). distributions of the interactions required to address problems in four datasets; b). solving rates of three types of reasoning path: “No Rollback” — thoughts receive no rollbacks, “Out Rollback” — rollbacks triggered by mistaken thoughts, and “In Rollback” — thoughts derive from rollbacks; c). the reduction of failure rates due to the rollbacks in the first step of Game of 24, where $[0, 1, 3, 5, 7]$ denotes the number of rollbacks that cause the first step and $2 \rightarrow 0$ means the first step derives from the rollback from the 2-th step.

Concerns. First, TR contributes less to the performance enhancement when the LLMs inherently do not have solid ability. Especially in AQUA-RAT and MATH of Table 1, GPT-3.5-turbo with TR is 2.91 and 3.25 higher than PHP+C-CoT, which are significantly lower than these under GPT-4. Likewise, in Table 2, the solving rate of Llama2-70b with TR is around 4% higher than Llama2-70b but costs more interactions. Therefore, the performance of TR depends on LLMs’ abilities to perform correct error analysis and understand the experience in the prompt. Second, to address harder problems, LLMs with TR tend to build over-complex thought structures, which generally contain more than 100 thoughts. The main reason is that a rollback generated by bad thoughts identified by LLMs or a mistaken rollback caused by hallucinations of LLMs leads to one more reasoning path. This appears frequently in challenging tasks; thus, LLMs self-organize a large-scale thought structure toward solutions. We present more visible examples in Figure 4, Figure 9, and Figure 10 of the Appendix.

5.2. Main Insights

The notable performance enhancement of TR in terms of both solving rate and interaction cost shows the insight that adaptive adjusting thoughts supported by the rollback of thoughts during reasoning is core to the success of LLMs in complex mathematical reasoning. In addition, we can gain three more insights.

Experience accumulation of error analysis from intermediate thoughts is better than that obtained by analyzing the whole reasoning path. Existing work (Huang et al., 2024) pointed out that LLMs are unable to revise reasoning based on the outcome analysis, which gives feedback on the final reasoning. Thus, BoT (Sijia et al., 2024), which relies on outcome analysis, had to embrace more careful-selected outcome analysis to prompt LLMs. Our TR opens a new direction of relying on process analysis, which provides error

analysis for each intermediate reasoning step (rollback-by-rollback verification), to revise thought adaptively during reasoning. Besides, with the rollback of thought, outcome analysis is a special case of process analysis when analysis is used not to re-do reasoning but to adjust previous thought to create a correct reasoning path.

Experience-guided Solution Ensemble is critical to the effectiveness of trial-and-error reasoning. After stopping reasoning, LLMs with TR yield K reasoning paths due to the adaptive exploration. Each reasoning path caused by one rollback of TR can be regarded as a trial for addressing the problem. When LLMs frequently make mistakes and occur “hallucinations”, the solution obtained in any trial may not be correct. Since TR exploits the error analysis of each *incoming rollback* as experience to prompt LLMs, a solution from the reasoning path with more *incoming rollback* is more acceptable. Therefore, we should ensemble these solutions by filtering out ones with limited experiences or many bad thoughts. As shown by comparisons between TR and TR + W-Voting in Table 1, Table 2, Table 3 and Figure 3 (b), such an experience-guided solution ensemble is critical.

Weak LLMs may not identify multiple targets mentioned in the prompt. Including CoT examples in the prompt improves the solving rate of LLMs with TR, as shown in the GPT-4 column of Table 1. However, for the more challenging AQUA-RAT and MATH datasets in the GPT-3.5-turbo column, adding CoT causes a significant performance decrease. We argue that it may be hard for the weak LLMs to understand and distinguish instructions with different targets in the prompt. For example, the instruction of CoT examples emphasizes how to follow demonstrations, while the prompt with experiences in TR focuses on how to avoid given errors. Weak LLMs may not benefit from the enhanced prompt containing these two different guidances, especially when the reasoning is complex.

Table 4. Evaluating the token cost when using GPT-4 with TR to address questions from MATH and TheoremQA. The ratios in the table indicate how many times the token costs of various approaches exceed the token cost of ZeroShot CoT. Δ represents the increase in the problem-solving rate of various approaches compared to ZeroShot CoT. We also provide the average and standard deviation values, expressed as mean \pm std, for the tokens required to prompt LLMs and those generated by LLMs in addressing a single question from two challenging datasets.

Method	MATH					TheoremQA				
	Generate tokens	Ratios	Prompt tokens	Ratios	Δ	Generate tokens	Ratios	Prompt tokens	Ratios	Δ
ZeroShot CoT	212.3 \pm 152.6	1	110.3 \pm 25.5	1	0	240.8 \pm 100.5	1	121.2 \pm 43.1	1	0
CoT _s	441.3 \pm 310.7	2.08	4611.4 \pm 1952.2	41.8	4.2	-	-	-	-	-
Chain Reasoning	603.2 \pm 244.7	2.84	2657.5 \pm 1187	24.1	0.74	976.3 \pm 1056.4	4.05	4614.6 \pm 7632.1	38.07	-7.25
ToT Reasoning	1569.8 \pm 474.7	7.39	9963.7 \pm 4487.5	90.33	3.3	1886 \pm 982.5	7.83	11982 \pm 8872.2	98.86	-5.5
TR + W-Voting	7484.9 \pm 5873.9	35.26	46904.9 \pm 37980.8	425.25	27.19	6762.9 \pm 6513	28.09	43444.8 \pm 49391.2	358.46	13

5.3. Token Cost Analysis

As shown in Table 4, when addressing questions from challenging datasets, the token cost of GPT-4 with TR is significantly higher than baseline approaches. Specifically, on the MATH dataset, the TR approach, on average, generates 35.26 times more tokens and requires 425.25 times more prompt tokens than the ZeroShot CoT. The corresponding ratios in the TheoremQA dataset are 28.09 and 358.46. This resource-intensive nature of our proposed TR derives from continuously identifying the errors and appending the prompt with error analysis during reasoning.

However, these additional operations and the high token cost are necessary as hallucinations of LLMs frequently appear. First, since numerous erroneous thoughts are generated during reasoning, consistently identifying and revising them is crucial to ensuring the correctness of the answer. Second, in many cases, the error analysis of LLMs is invalid and even incorrect due to the inevitable hallucinations. Thus, accumulating error analysis derived from different reasoning paths decreases the negative impact of the flawed analysis on thought revisions. Third, many reasoning paths of the thought structure derive from rollbacks triggered by erroneous feedback from LLMs. Since mistaken rollbacks cannot be identified, the TR approach retains all generated paths and ultimately employs majority voting to enhance reliability.

Therefore, we conclude that there is a trade-off between the high token cost and the problem-solving rate. On the one hand, since the TR approach requires many tokens to address a single question, its application may be limited for users with insufficient resources. On the other hand, compared to zero-shot GPT-4, GPT-4 with TR gains 27.19% and 13% improvements in solving rates on MATH and TheoremQA datasets. When users prioritize problem-solving rates, integrating GPT-4 with the TR approach ensures its applicability in many challenging scenarios.

6. Concluding Remarks

In this paper, we proposed Thought Rollback (TR), an effective reasoning framework supported by the rollback of thoughts that allows LLMs to perform adaptive reasoning to solve challenging problems. Without relying on human annotations and specific thought structure designs for reasoning, LLMs with TR can progressively self-organize and revise thoughts based on trial-and-error experiences until reaching a correct solution for various tasks. The *rollback controller* and *prompt enhancer*, together with the experience-guided weights majority voting, make TR achieve the state-of-the-art solving rate in many mathematical and multi-task reasoning datasets while maintaining a lower cost than the alternative leading approaches. We hope this work could shed light on the adaptive reasoning in LLMs toward addressing challenging tasks, especially when mathematical problems are involved.

Impact Statement

Large language models (LLMs) can break a complex task into manageable subproblems and solve them through step-by-step reasoning. TR, a lightweight framework, guarantees the reliability of LLMs’ multi-step reasoning under hallucinations, thus extending their applications to a wider range of tasks. Furthermore, compared to the BOT that performs outcome analysis, TR built upon prompting the LLMs with feedback from process analysis is more effective and significantly reduces the interaction cost. This may open a research direction emphasizing the importance of exploiting feedback during the step-by-step reasoning of LLMs. In addition, the plug-and-play nature of TR allows other approaches, such as CR, to involve the thought rollback mechanism to improve performance further. Ultimately, thanks to the expansion in the context window of LLMs and the decreasing token prices, the token cost of TR may not be a major concern.

References

- An, S., Ma, Z., Lin, Z., Zheng, N., Lou, J.-G., and Chen, W. Learning from mistakes makes llm better reasoner. *arXiv preprint arXiv:2310.20689*, 2023.
- Besta, M., Blach, N., Kubicek, A., Gerstenberger, R., Gianinazzi, L., Gajda, J., Lehmann, T., Podstawski, M., Niewiadomski, H., Nyczyk, P., et al. Graph of thoughts: Solving elaborate problems with large language models. *arXiv preprint arXiv:2308.09687*, 2023.
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020.
- Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.
- Chen, W., Ma, X., Wang, X., and Cohen, W. W. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, 2023a.
- Chen, W., Yin, M., Ku, M., Lu, P., Wan, Y., Ma, X., Xu, J., Wang, X., and Xia, T. Theoremqa: A theorem-driven question answering dataset. In *Proc. Conference on Empirical Methods in Natural Language Processing*, 2023b.
- Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- Fu, Y., Peng, H., Sabharwal, A., Clark, P., and Khot, T. Complexity-based prompting for multi-step reasoning. In *Proc. International Conference on Learning Representations*, 2023.
- Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., and Neubig, G. Pal: Program-aided language models. In *International Conference on Machine Learning*, pp. 10764–10799. PMLR, 2023.
- Hagberg, A. A., Schult, D. A., and Swart, P. J. Exploring network structure, dynamics, and function using networkx. In *Proc. 7th Python in Science Conference*, pp. 11–15, 2008.
- Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. In *Proc. International Conference on Learning Representations*, 2021a.
- Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021b.
- Huang, J., Chen, X., Mishra, S., Zheng, H. S., Yu, A. W., Song, X., and Zhou, D. Large language models cannot self-correct reasoning yet. In *Proc. International Conference on Learning Representations*, 2024.
- Jiang, M., Ruan, Y., Huang, S., Liao, S., Pitis, S., Grosse, R. B., and Ba, J. Calibrating language models via augmented prompt ensembles. In *International Conference on Machine Learning*, 2023.
- Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pp. 22199–22213, 2022.
- Kong, A., Zhao, S., Chen, H., Li, Q., Qin, Y., Sun, R., and Zhou, X. Better zero-shot reasoning with role-play prompting. *arXiv preprint arXiv:2308.07702*, 2023.
- Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let’s verify step by step. In *Proc. International Conference on Learning Representations*, 2024.
- Ling, W., Yogatama, D., Dyer, C., and Blunsom, P. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*, 2017.
- Ling, Z., Fang, Y., Li, X., Huang, Z., Lee, M., Memisevic, R., and Su, H. Deductive verification of chain-of-thought reasoning. In *Advances in Neural Information Processing Systems*, 2023.
- Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.
- Luo, L., Li, Y.-F., Haffari, G., and Pan, S. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *Proc. International Conference on Learning Representations*, 2024.
- Lyu, Q., Havaldar, S., Stein, A., Zhang, L., Rao, D., Wong, E., Apidianaki, M., and Callison-Burch, C. Faithful chain-of-thought reasoning. In *Proc. IJCNLP-AACL*, 2023.
- Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegrefe, S., Alon, U., Dziri, N., Prabhunoye, S., Yang, Y., et al. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*, 2023.

- OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- Patel, A., Bhattamishra, S., and Goyal, N. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*, 2021.
- Qin, C., Zhang, A., Zhang, Z., Chen, J., Yasunaga, M., and Yang, D. Is chatgpt a general-purpose natural language processing task solver? In *Proc. Conference on Empirical Methods in Natural Language Processing*, 2023.
- Sijia, C., Baochun, L., and Niu, D. Boosting of thoughts: Trial-and-error problem solving with large language models. In *Proc. International Conference on Learning Representations*, 2024.
- Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and finetuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. In *Proc. International Conference on Learning Representations*, 2022.
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022.
- Weng, Y., Zhu, M., Xia, F., Li, B., He, S., Liu, S., Sun, B., Liu, K., and Zhao, J. Large language models are better reasoners with self-verification. In *Proc. Conference on Empirical Methods in Natural Language Processing*, pp. 2550–2575, 2023.
- Wu, Z., Jiang, M., and Shen, C. Get an a in math: Progressive rectification prompting. In *Proc. AAAI Conference on Artificial Intelligence*, 2024.
- Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, 2023.
- Yu, J., He, R., and Ying, R. Thought propagation: An analogical approach to complex reasoning with large language models. In *Proc. International Conference on Learning Representations*, 2024.
- Zhang, Y., Yang, J., Yuan, Y., and Yao, A. C.-C. Cumulative reasoning with large language models. *arXiv preprint arXiv:2308.04371*, 2023.
- Zhang, Z., Zhang, A., Li, M., and Smola, A. Automatic chain of thought prompting in large language models. In *Proc. International Conference on Learning Representations*, 2022.
- Zhao, X., Xie, Y., Kawaguchi, K., He, J., and Xie, Q. Automatic model selection with large language models for reasoning. In *Proc. Conference on Empirical Methods in Natural Language Processing*, 2023.
- Zheng, C., Liu, Z., Xie, E., Li, Z., and Li, Y. Progressive-hint prompting improves reasoning in large language models. *arXiv preprint arXiv:2304.09797*, 2023.
- Zhou, A., Wang, K., Lu, Z., Shi, W., Luo, S., Qin, Z., Lu, S., Jia, A., Song, L., Zhan, M., et al. Solving challenging math word problems using gpt-4 code interpreter with code-based self-verification. In *Proc. International Conference on Learning Representations*, 2024.

A. Supplement to Figures of the Main Paper

Figure 4 is the full version of the partial one shown in Figure 1. This shows a complex thought structure built by GPT-4 with TR to address a challenging question from MATH dataset. It shows that GPT-4 with TR tends to build a large-scale thought structure by iteratively rolling back thoughts during reasoning. Figure 4 demonstrates the details about how TR exploits *rollback controller* and *prompt enhancer* to create a new and correct reasoning path after analyzing the bad thoughts.

Specifically, from the details presented in Figure 5, we can observe what TR shown in Figure 2 does after reaching the thought. GPT-4 with TR starts from generating a first thought $N - 1S - 1$ based on a simple and zero-shot prompt Q . Then, **Rollback of thoughts** follow the process in Figure 2. First, *rollback controller* analyzes the reasoning path $N - 1 \rightarrow N - 2 \rightarrow N - 3 \rightarrow N - 5 \rightarrow N - 6$ to output the error analysis presented in “N-6, S-5 **error analysis:**”. This analysis shows that reasoning steps $N - 5S - 4$ and $N - 6S - 5$ are bad thoughts. According to our discussion in subsection 4.2, *rollback controller* allows the LLM to roll back to the thought $N - 3S - 3$, which is one step before the first bad thought $N - 5S - 4$. Then, *prompt enhancer* accumulates the error analysis as experience in the prompt, as shown in “**N-3, S-3 to N-7 S-4 Prompt**” which includes the “##### The 0-th Experience with Analysis #####”. As a result, by avoiding making similar mistakes mentioned by experience, LLM is able to generate a new thought $N - 7S - 4$ from the chosen thought $N - 3S - 3$. Therefore, “hallucinations” that occur in thought or analysis of LLM may not influence reasoning due to the continuous thought revision guaranteed by the iterative rollbacks during reasoning. As can be seen from $N - 9S - 6$, the final solution is revised to be a correct answer 737. Additionally, we can also observe that each rollback will lead to a new reasoning path with the experience from the corresponding error analysis. Thus, two rollbacks $N - 6 \rightarrow N - 3$ and $N - 3 \rightarrow N - 2$ of GPT-4 with TR adaptively create two new reasoning paths.

B. Reproducibility of Thought Rollback Framework

B.1. Source Code

One can also access the source code under the *examples/ThoughtRollback* folder in the *code*. The implementation is based on the *llmpebase* library. The code is written in Python and imports the datasets from Hugging Face to build PyTorch’s data loader.

Also, the source code for Chain Of Thought, Chain Reasoning, ToT reasoning, and GoT reasoning that we mention in experiments are available in the *examples/ChainOfThought*, *examples/ChainReasoning*, *examples/TreeReasoning*, and *examples/GraphReasoning* folders, respectively.

All configuration files used to conduct the experiments are provided in the *configs/* folder in the *code*. About how to run, please read the README.md under the *examples/ThoughtRollback* folder.

B.2. Locations of Generated Thoughts and Reasoning Details

Our code will automatically save the generated thoughts and reasoning details under one folder of *LLMPE* in the root directory. The direct results are placed under the *LLMPE/results* while the corresponding visible thought structures are stored in *LLMPE/visualizations*. Then, their sub-folder name will represent the setting of the configuration, such as “TRReasoning_gpt-4_zeroshot_cot_MATH”, where “TRReasoning” is the name of our Thought Rollback. Eventually, as shown in Figure 6, you can access the sample by index in *thought_structure_** while reading the results in *llm_records*. The thought structure for reasoning will be saved in the *.json* format, and the visualizations will be in the *.pdf* format.

B.3. Prompts

This subsection presents the basic prompts used in our implementation of Thought Rollback framework.

System prompt for thought generation: *You possess expertise in solving mathematical problems through a systematic, step-by-step reasoning process during which you are dedicated to preventing repeating any errors analyzed in experiences. Your objective is to address the question using a series of reasoning steps delivered in multiple responses, with each response containing one reasoning step. It is crucial to avoid repeating errors mentioned in the given experiences. Begin by reading the provided reasoning steps and then proceed to generate the most appropriate next step in the response, ensuring that the logical progression steadily leads towards a solution.*

System prompt for reasoning analysis: *You are a mathematician specializing in checking and analyzing the reasoning*

process containing multiple intermediate reasoning steps proposed to address a math question. Please check the correctness of the overall reasoning logic and each reasoning step regarding mathematical logic and rationality.

Prompt \mathbb{I} for the next thought generation:

Answer the question about the problem $\{Problem\ Name\}$. After getting the final solution, place it after the sentence 'The final solution is' for readability.
 $\backslash n \backslash n$ Experience containing previously made mistakes: $\backslash n \backslash n$ ##### $\{Experiences\}$ ##### $\backslash n \backslash n$ Consider the analysis in the above experience to avoid making similar mistakes during reasoning for the question.
 $\backslash n \backslash n \backslash n$ Question: $\{Question\}$ $\backslash n \backslash n$ Answer: Let's think step by step. Let's focus on carefully generating the next possible reasoning step for reasoning steps below.
 $\backslash n \backslash n \backslash n \backslash n$ $\{Existing\ Reasoning\ Steps\}$ $\backslash n \backslash n \backslash n$ For reasoning steps within , please generate their best next step containing analysis and the corresponding mathematical expression.

where the $\{Problem\ Name\}$ contains what question to solve, such as "Multiplication", $\{Experiences\}$ present the accumulated experiences $A_{z_{0...q-1}}^{\chi^{(\cdot)}}$ discussed in subsection 4.3, $\{Question\}$ is the given question description and finally $\{Experiences\}$ is a placeholder to be replaced by the preceding chain of thoughts $z_{0...n-1}$.

Prompt \mathbb{I}_R for the error analysis:

Analyze the reasoning steps proposed for the question about the problem $\{Problem\ Name\}$.
 $\backslash n$ Question: $\{Question\}$ $\backslash n$ Toward addressing the given question, below is a reasoning process containing $\{Number\ of\ Steps\}$ steps: $\backslash n \backslash n \backslash n$ $\{Existing\ Reasoning\ Steps\}$ $\backslash n \backslash n \backslash n$ Double-check the reasoning process within , please analyze its overall and each step's correctness by checking whether they are mathematical logic and rationality. Please report an error when any step does not contain a clear mathematical expression. Output empty string when no steps are given.
 $\backslash n$

where the $\{Number\ of\ Steps\}$ presents the number of current reasoning steps, i.e., n for a reasoning path $z_{0...n}$.

B.4. Basic Engineering Settings for TR

During implementation, we set the upper bound U of the incoming rollback for one thought to be 3, making no more than 3 rollbacks use this thought as the destination. Besides, we utilize the depth-first search algorithm to find the current growing thought. As mentioned in the subsection 4.4, the reasoning path with more incoming rollback is more important. Therefore, for this depth-first search algorithm, we assign higher priority to the reasoning path with a larger number of incoming rollbacks. As LLMs with TR first generate subsequent thoughts for these reasoning paths, the corresponding solution will be explored in advance. Such a mechanism increases the possibility of getting better answers.

Apart from these basic settings, our implementation also includes some engineering tricks. First, during reasoning, once a reasoning path causes outgoing rollbacks more than 5 times, it will be ignored in the subsequent reasoning. Second, we do not allow one thought to cause more than 3 outgoing rollbacks to avoid the case that LLMs repeatedly identify a bad thought and trigger rollbacks for it. Third, to increase the speed of reasoning, after noticing that different reasoning paths are independent of each other, we achieve parallel running, which makes LLM generate thoughts for all reasoning paths simultaneously. Therefore, once a new reasoning path is created, one process will be created to make LLM work on the path without blocking others. This can increase the reasoning speed from the scale of minutes to seconds.

C. Discussion: Insights Gained from the TR Framework

We argue that the outstanding performance of TR is attributed to three insights.

First, as mentioned by the work (Lightman et al., 2024), compared to outcome supervision, which provides feedback for a final result, process supervision, which provides feedback for each intermediate reasoning step, is more important to guarantee the reliable reasoning of LLMs. In summary, process supervision significantly outperforms outcome supervision. Actually, TR roughly belongs to process supervision because it performs rollback-by-rollback error analysis during reasoning and continues to revise thoughts based on the accumulation of experience. Thus, the reasoning of LLMs can be adjusted adaptively during reasoning rather than after the results are obtained. However, BoT (Sijia et al., 2024) belongs to outcome supervision as it only collects error analysis after reasoning.

Second, the work (Huang et al., 2024) figured out that the outcome analysis is sometimes invalid as LLMs cannot use feedback to revise thoughts to improve the solving rate. As discussed in subsection 3.2 of our paper, this may be because when there are many errors in the intermediate reasoning steps, capturing the source mistake and performing useful analysis

is challenging. Thus, BoT (Sijja et al., 2024) has to do many tree ensembles and design a complex boosting mechanism to mine effective error analysis to drive LLM’s reasoning revision. On the contrary, TR performs thought rollback triggered by continuous error analysis during reasoning. Thus, once the error is identified, LLMs can directly revise or adjust the reasoning based on timely accumulated experience in the prompt.

Third, prompting LLMs with a long input context may cause the degradation of reasoning performance. As pointed out by the work (Liu et al., 2024), LLMs do not fairly utilize all the information in the prompt but focus more on the content at the beginning or end of the input context. For BoT, the prompt tends to become extremely long due to the continuous collection of generated reasoning steps and their error analysis over iterations, particularly since the error analysis focuses on the whole reasoning chain. For instance, in the 10-th iteration, the prompt contains ten long contents, each containing all reasoning steps and step-wise analysis. Some of them may not even be correct. However, our TR accumulates experience during reasoning; thus, each experience only contains analysis on very few intermediate steps, generally leading to a quick revision after the rollback. For instance, LLMs with TR can easily generate a correct second step based on the experience “Reasoning step 1: ... correct. Reasoning step 2: ... wrong because....”. However, the BoT’s experience “Reasoning step1:correct. Reasoning step 2: ... wrong because.... . Reasoning step 3: ... wrong because..... Reasoning step 4: ... wrong because.... . Reasoning step 5: ... wrong because....” may just mislead the LLM.

Finally, regarding the time complexity of the TR framework, LLMs with TR incrementally construct the thought structure from #node 0 to #node n . By focusing on the leading term and disregarding constants and lower order terms, the worst-case time complexity of TR is determined to be $\mathcal{O}(n^2)$.

C.1. Source of Experimental Results

In Table 5 and Table 6, we collect experimental results of GPT-4 and GPT-3.5-turbo on various settings. We especially show the corresponding work that reports the results.

Table 5. Source of experiment results of GSM8K. Methods are CSV (Zhou et al., 2024), PHP (Zheng et al., 2023) with GPT-3.5-turbo-, Model Selection (MS) (Zhao et al., 2023), PAL (Gao et al., 2023), PLAY (Kong et al., 2023), Faithful (Lyu et al., 2023), Exps (Bubeck et al., 2023), PoT (Chen et al., 2023a), IG (Qin et al., 2023), BoT (Sijja et al., 2024).

Source	GPT-4					GPT-3.5-turbo				
	ZeroShot	FewShot	ZeroShot-CoT	CoT	Complex-CoT	ZeroShot	FewShot	ZeroShot-CoT	CoT	Complex-CoT
CSV	Code	-	-	92.0 ₅	-	-	57.1 ₅	-	-	-
	Code+CSV	92.9, 94.9 _{sc5}	-	-	-	-	-	-	-	-
PHP	+PHP	-	-	-	94.9 ₈	-	-	-	-	82.8 ₈
		-	-	-	95.5 ₈	-	-	-	-	85.1 ₈
MS	PAL	-	-	94.6 ₅ , 95.6 _{5,sc5} , 95.6 _{5,sc15}	-	-	-	80.8 ₅ , 85.4 _{5,sc5} , 87.4 _{5,sc15}	-	-
	Ours	-	-	94.0 ₅ , 94.7 _{5,sc5} , 95.5 _{5,sc15}	-	-	-	79.2 ₅ , 80.9 _{5,sc5} , 82.4 _{5,sc15}	-	-
PLAY	Role	-	-	-	-	76.0	-	79.6	76.9	-
		-	-	-	-	78.2	-	-	-	-
Faithful	Faithful	46.9	-	64.9 ₈	-	-	-	-	-	-
		-	-	95 ₈	-	-	-	-	-	-
Exps		87.1	-	-	-	-	-	-	-	-
PoT*		-	-	-	-	76.3	-	-	-	-
IG		-	-	-	-	23.8	-	78.9	-	-
BoT		87.1	-	89.6	92 ₈	94.9	-	-	-	-
	BoT	-	-	97.1	98.7 ₈	-	-	-	-	-

D. Examples of GPT-4 with TR in GSM8K

In Figure 7, we present a simple reasoning performed by GPT-4 with TR. As no bad thoughts are identified during reasoning, GPT-4 with TR directly performs correct reasoning toward a correct solution. This simple example aims to give an overview of 1). how multi-step reasoning with multi-step prompts work; 2). how to prompt LLMs to generate the next thought, such as $N - 2 S - 2 \rightarrow N - 3 S - 3$; and 3) how LLMs with TR are able to perform normal reasoning when no rollback is triggered. Besides, as discussed in subsection 4.1, LLMs with TR start from a zero-shot prompt containing only the question and task information.

Then, Figure 8 shows a more complex reasoning process conducted by GPT-4 with TR. In this case, *rollback controller* triggers 5 rollbacks during reasoning, leading to 8 different solutions. After operating the experience-guided solution ensemble, we get 6 as the final answer, which is correct. LLMs with TR first generate one thought $N - 1 S - 1$. Then, *rollback controller* exploits LLMs to analyze the current reasoning path $N - 0 S - 1$, $N - 1 S - 1$ and $N - 2 S - 2$ and thus identifies the error in $N - 2 S - 2$. This triggers a rollback $N - 2 S - 2 \rightarrow N - 1 S - 1$, leading to a new reasoning

Table 6. Source of experiment results of SVAMP, AQuA-RAT, MATH, TheoremQA and MMLU. They are CSV (Zhou et al., 2024), PHP (Zheng et al., 2023), Model Selection (MS) (Zhao et al., 2023), PAL (Gao et al., 2023), PLAY (Kong et al., 2023), Faithful (Lyu et al., 2023), Exps (Bubeck et al., 2023), PoT (Chen et al., 2023a), IG (Qin et al., 2023), BoT (Sijia et al., 2024), CR (Zhang et al., 2023), TheoremQA (Chen et al., 2023b) and GPT4-report (OpenAI, 2023).

Source		SVAMP									
		GPT-4			GPT-3.5-turbo			GPT-3.5-turbo			
		ZeroShot	FewShot	ZeroShot-CoT	CoT	Complex-CoT	ZeroShot	FewShot	ZeroShot-CoT	CoT	Complex-CoT
PHP	+PHP	-	-	-	-	90.5 ₈	-	-	-	-	81.0 ₈
		-	-	-	-	91.9 ₈	-	-	-	-	83.1 ₈
MS	PAL	-	-	-	91.9 ₅	-	-	-	-	83 ₅	-
	Ours	-	-	-	92.2 ₅	-	-	-	-	80.3 ₅	-
PLAY	Role	-	-	-	-	-	75.3	-	76.3	84.3 ₅	-
		-	-	-	-	-	83.8	-	-	82.2	-
Faithful	Faithful	-	88.4	-	80 ₈	-	-	-	-	-	-
PoT*		-	-	-	95.4 ₈	-	-	-	-	-	-
IG		-	-	-	-	-	88.2	-	-	-	-
BoT		68.7	-	74.3	77.6 ₈	90.5 ₈	-	-	77.5	-	-
	BoT	-	-	92.7	94.9 ₈	-	-	-	-	-	-

Source		AQuA-RAT									
		GPT-4			GPT-3.5-turbo			GPT-3.5-turbo			
		ZeroShot	FewShot	ZeroShot-CoT	CoT	Complex-CoT	ZeroShot	FewShot	ZeroShot-CoT	CoT	Complex-CoT
PHP	+PHP	-	-	-	-	77.5 ₈	-	-	-	-	57.4 ₈
		-	-	-	-	79.9 ₈	-	-	-	-	60.6 ₈
PLAY	Role	-	-	-	-	-	53.5	-	53.9	59.4	-
		-	-	-	-	-	63.8	-	-	-	-
Faithful	Faithful	50.4	-	-	75.2 ₈	-	-	-	-	-	-
PoT*		-	-	-	73.6 ₈	-	-	-	-	-	-
IG		-	-	-	72.4	-	-	-	-	-	-
BoT		40.6	-	73.2	74 ₈	77.5	28.0	-	53.5	-	-
	BoT	-	-	81.4	84.9 ₈	-	-	-	-	-	-

Source		MATH									
		GPT-4			GPT-3.5-turbo			GPT-3.5-turbo			
		ZeroShot	FewShot	ZeroShot-CoT	CoT	Complex-CoT	ZeroShot	FewShot	ZeroShot-CoT	CoT	Complex-CoT
CSV	Code	42.2	-	-	-	50.36 ₈	-	-	-	-	34.12 ₈
	Code+CSV	69.69, 79.88 _{ec16}	-	-	-	-	-	-	-	-	-
PHP	+PHP	73.54, 83.54 _{ec16}	-	-	-	42.5 ₈	50.36 ₈	-	-	-	34.12 ₈
		84.32 _{ec16}	-	-	-	53.9 ₈	-	-	-	-	36.5 ₈
BoT	BoT	42.5	-	47.7	48.93 ₈	50.4 ₈	-	-	-	-	-
		-	-	62.5	66.3 ₈	-	-	-	-	-	-
CR ₅₀₀	CR	-	-	-	54.2 ₄	-	-	-	-	-	-
	PHP+CR	-	-	-	58 ₄	-	-	-	-	-	-

Source		TheoremQA									
		GPT-4			GPT-3.5-turbo			GPT-3.5-turbo			
		ZeroShot	FewShot	ZeroShot-CoT	CoT	Complex-CoT	ZeroShot	FewShot	ZeroShot-CoT	CoT	Complex-CoT
TheoremQA	PoT	-	-	-	43.8	-	-	-	-	30.2, 30.8 _{theorem}	-
		-	52.4	-	-	-	35.6, 35.8 _{theorem}	-	-	-	-

Source		MMLU									
		GPT-4			GPT-3.5-turbo			GPT-3.5-turbo			
		ZeroShot	FewShot	ZeroShot-CoT	CoT	Complex-CoT	ZeroShot	FewShot	ZeroShot-CoT	CoT	Complex-CoT
GPT4-report	BoT	86.5	86.4	90.86	93.42 ₅	-	70	-	-	-	-
		-	-	-	-	-	-	-	-	-	-

path $N - 1 S - 1 \rightarrow N - 3 S - 2$ in which the prompt includes the error analysis, thereby revising the thought to gain a final correct solution $N - 9 S - 4$.

As there are 3 incoming rollbacks for the thought $N - 10 S - 2$, the corresponding three different error analyses are accumulated as shown by the **N-10 S-2 Experience Accumulation**. *prompt enhancer* include these error analyses as experiences in the prompt to guide LLMs to produce correct thoughts. For instance, **N-10 S-2** \rightarrow **N-17 S-3 Prompt** shows that GPT-4 generates the thought $N - 17 S - 3$ from the thought $N - 10 S - 2$ with the prompt that accumulates two experiences.

Eventually, GPT-4 with TR adaptively builds a thought structure towards generating solutions that, most of which are correct due to the continuous thought revisions via rollback of thoughts.

E. Examples of GPT-4 with TR in MATH

Limited by space, we store the detailed experimental result and visualization files of Figure 9 in the folder *MATH-example-1* of the supplementary. Thus, we present the question, the thoughts of correct solutions, and the values of all solutions. Specifically, in response to our discussion in section 4, we present how GPT-4 with TR is able to identify the bad thought $N - 34 S - 6$ and thus the *rollback controller* gets the $N - 34 S - 6$ **error analysis**. The triggered rollback $N - 34 S - 6 \rightarrow N - 32 S - 4$ leads to a new reasoning path $N - 32 S - 4 \rightarrow N - 35 S - 5$, which generates a correct thought and gets

the correct answer 0.

We specifically utilize the example in Figure 10 to show how **Experience Accumulation** works in the TR framework. For the reasoning path from $N - 0 S - 0$ to $N - 17 S - 3$, there are 4 *incoming rollbacks*, including the rollback $N - 3 S - 2 \rightarrow N - 0 S - 0$, the rollback $N - 10 S - 3 \rightarrow N - 4 S - 1$, and the rollback $N - 15 S - 3 \rightarrow N - 11 S - 2$. As each rollback creates an error experience from one trial of the given question, *incoming rollbacks* lead to a series of experiences, as shown by **N-11 S-2 Experience Accumulation**. Therefore, to generate $N - 17 S - 3$ from $N - 11 S - 2$, the *prompt enhancer* includes these error analyses as experiences in the prompt, as shown in **N-11 S-2 ; N-17 S-3 Prompt**. By learning from these experiences, LLMs are able to generate the correct thought $N - 17 S - 3$ and thus the correct answer 47. The detailed files of this example is presented in the folder *MATH-example-2* of the supplementary.

F. Examples of GPT-4 with TR in TheoremQA

GPT-4 with TR tends to build complex thought structures when reasoning with the challenging TheoremQA dataset (Chen et al., 2023b). As seen in Figure 11, the overall thought scale and the complexity of reasoning paths increase a lot compared to other examples.

Also, we show in Figure 12 that GPT-4 with TR can build even more complex thought structures.

In these two figures, we present the obtained reasoning path toward the correct answer. For example, the “Final solution of the reasoning path N-0 S-0 \rightarrow N-11 S-7:” in Figure 11 and the “Final solution of the reasoning path N-0 S-0 \rightarrow N-12 S-7:” in Figure 12 are the correct solutions obtained by GPT-4 with TR.

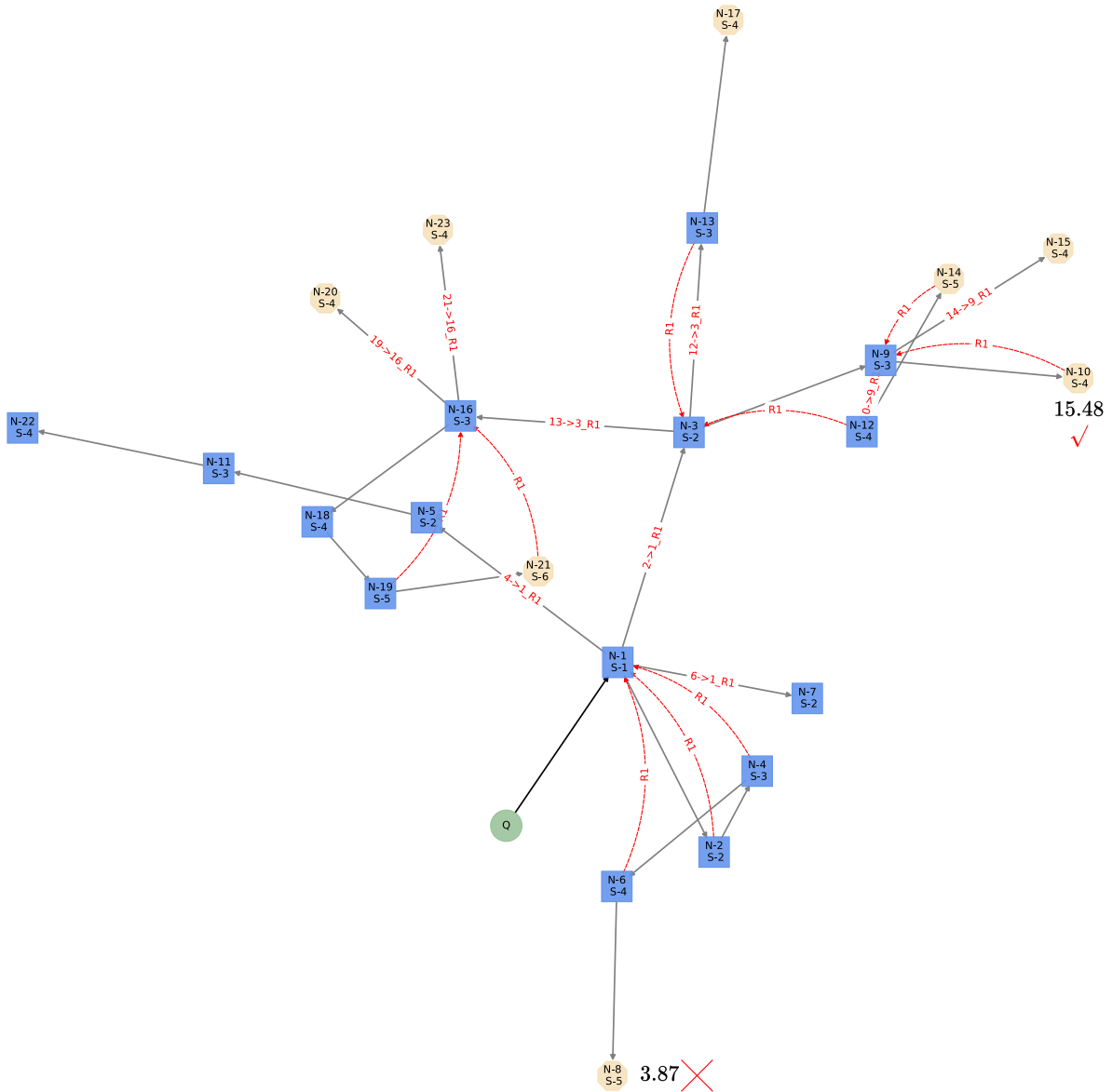


Figure 4. Complete thought structure of Figure 1 (c) built by GPT-4 with TR for the question from MATH dataset (Hendrycks et al., 2021b). This structure contains 23 nodes, i.e. 23 thoughts and leads to $K = 8$ reasoning paths towards solutions. It is plotted by based on the NetworkX lib (Hagberg et al., 2008) under the “fdp” layout.

Toward Adaptive Reasoning in Large Language Models with Thought Rollback

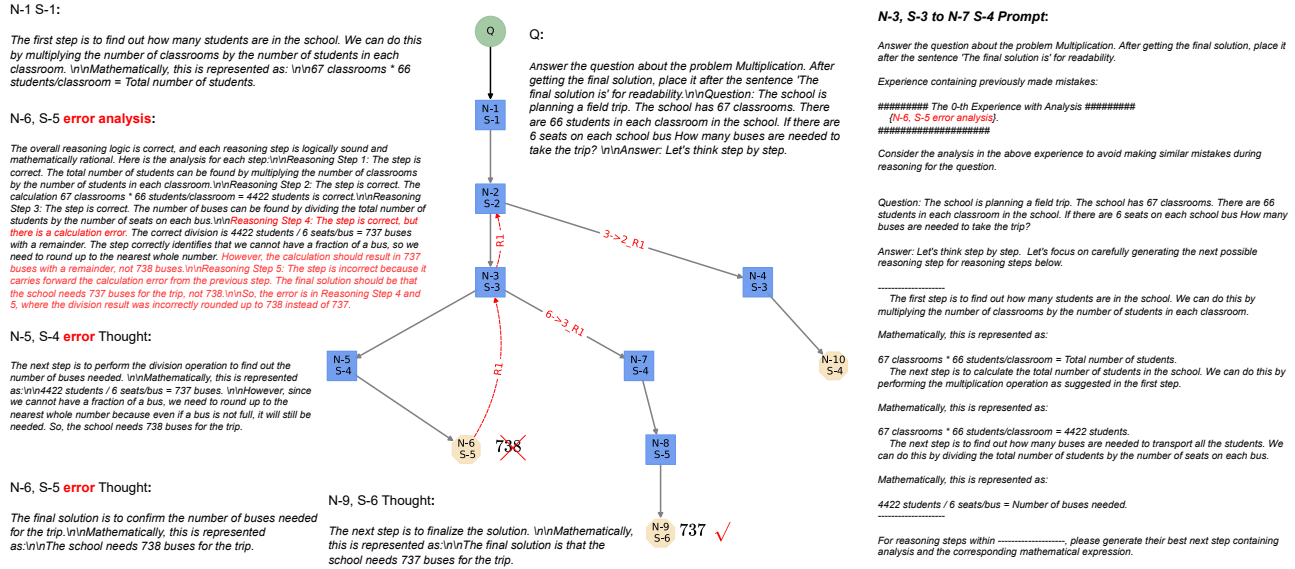


Figure 5. Complete thought structure of Figure 2 built by GPT-4 with TR for the question from SVAMP dataset (Patel et al., 2021). This structure contains 10 nodes, i.e. 10 thoughts and leads to $K = 3$ reasoning paths towards solutions. It is plotted by based on the NetworkX lib (Hagberg et al., 2008) under the “dot” layout.

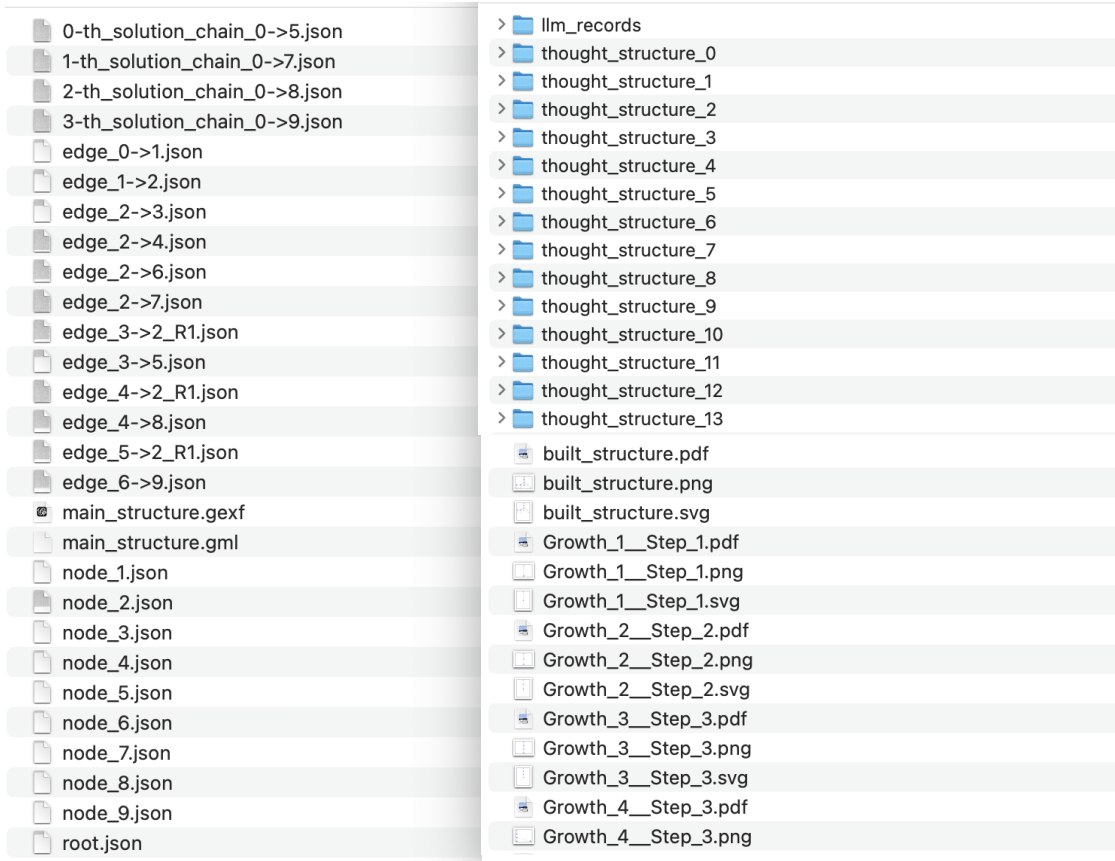
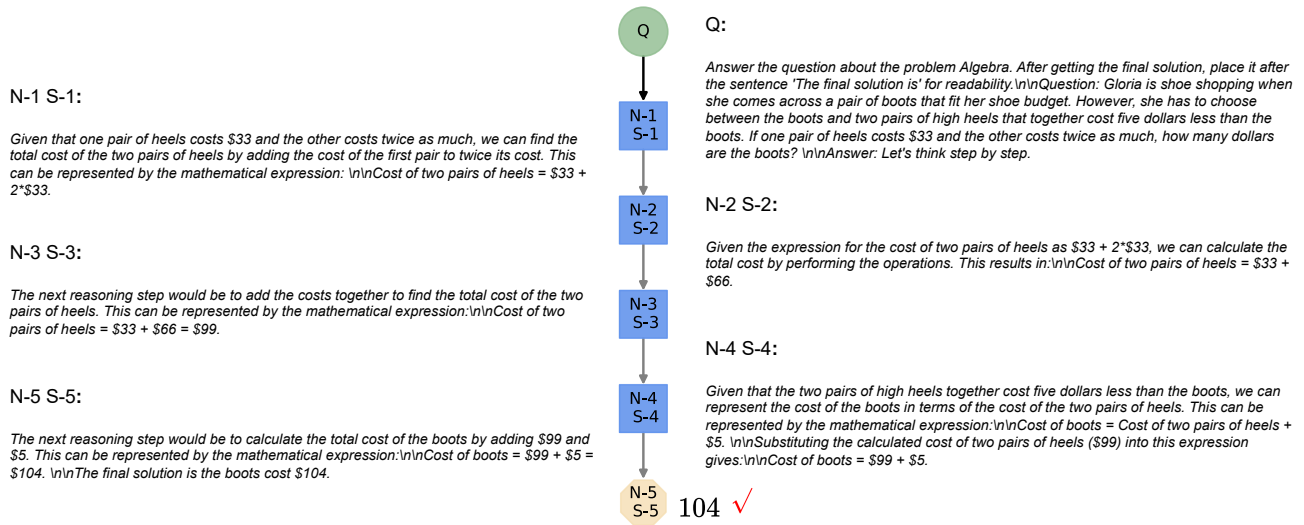


Figure 6. Illustrating of all results generated by LLMs with TR. The left sub-figure presents the details of the generated thought structure. The upper right sub-figure shows the files of obtained results, while the lower right sub-figure presents the visualizations.



N-2 S-2 -> N-3 S-3 Prompt:

Answer the question about the problem Algebra. After getting the final solution, place it after the sentence 'The final solution is' for readability.
 Question: Gloria is shoe shopping when she comes across a pair of boots that fit her shoe budget. However, she has to choose between the boots and two pairs of high heels that together cost five dollars less than the boots. If one pair of heels costs \$33 and the other costs twice as much, how many dollars are the boots?
 Answer: Let's think step by step. Let's focus on carefully generating the next possible reasoning step for reasoning steps below.

 Given that one pair of heels costs \$33 and the other costs twice as much, we can find the total cost of the two pairs of heels by adding the cost of the first pair to twice its cost. This can be represented by the mathematical expression:
 $\text{Cost of two pairs of heels} = \$33 + 2 * \$33.$
 Given the expression for the cost of two pairs of heels as $\$33 + 2 * \33 , we can calculate the total cost by performing the operations. This results in:
 $\text{Cost of two pairs of heels} = \$33 + \$66.$

 For reasoning steps within -----, please generate their best next step containing analysis and the corresponding mathematical expression.

Figure 7. A simple thought structure built by GPT-4 with TR for the question from GSM8K dataset (Cobbe et al., 2021). This structure contains 5 nodes, i.e. 5 thoughts and leads to $K = 1$ reasoning path towards one solution because no error is identified by the rollback controller with GPT-4 during reasoning.

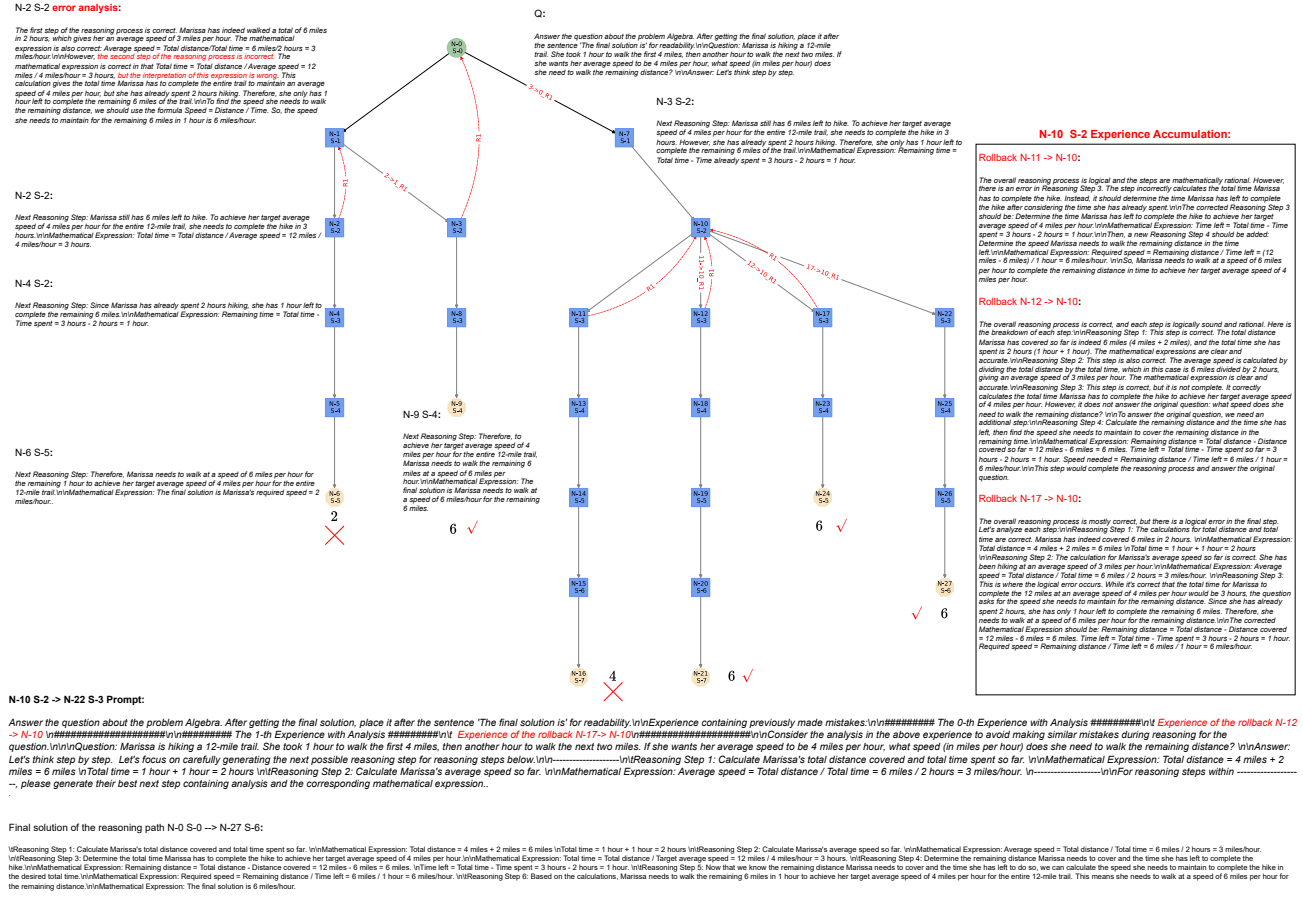


Figure 8. A slightly complex thought structure built by GPT-4 with TR for the question from GSM8K dataset (Cobbe et al., 2021). This structure contains 27 nodes, i.e. 27 thoughts and leads to $K = 8$ reasoning paths towards 8 solutions as 5 number of rollbacks are triggered by the rollback controller with GPT-4 during reasoning.

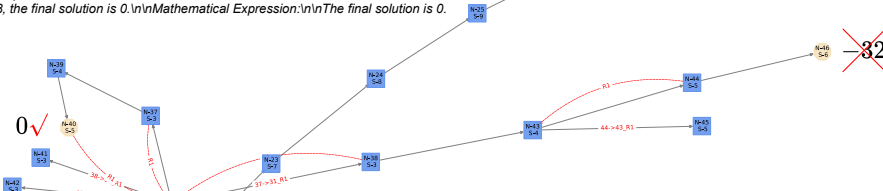
N-40 S-5:

Reasoning Step: \n\nNow, let's observe that the polynomial $p(x)$ has the same value at four different points: 55, 83, 204, and 232. This is a characteristic of periodic functions, not polynomials. However, we can take advantage of this observation by considering the polynomial modulo these differences. In other words, we can consider the values of $p(x)$ for x modulo 55, 83, 204, and 232. \n\nIf we look at the numbers 1 and 286 (the arguments of the terms that do not cancel out in the telescoping series), we can see that 286 is congruent to 1 modulo 55, 83, 204, and 232. This means that $p(286)$ should be equal to $p(1)$ because the polynomial has the same value at these congruent points. Therefore, the value of the expression $p(1) - p(286)$ should be zero. \n\nThe final solution is 0. \n\nMathematical Expression: \n\n $p(1) - p(286) = 0$.

$$2[a*(143^5/5 + 143^4/2 + a_2*143^3 + a_1*143^2 + a_0*143) + b*(143^4/4 + 143^3/2 + b_2*143^2 + b_1*143 + b_0) + c*(143^3/3 + 143^2/2 + c_1*143 + c_0) + d*(143^2/2 + 143) + 143^2] - 64.$$

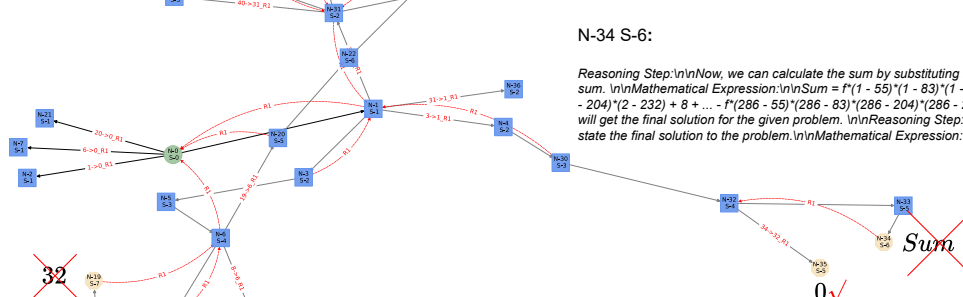
N-17 S-7:

Reasoning Step: \n\nSince all pairs of consecutive terms cancel each other out and the roots of the polynomial result in $p(x) = 8$, the final solution is 0. \n\nMathematical Expression: \n\nThe final solution is 0.



N-34 S-6:

Reasoning Step: \n\nNow, we can calculate the sum by substituting the value of f into the expression for the sum. \n\nMathematical Expression: \n\n $Sum = f(1 - 55)(1 - 83)(1 - 204)(1 - 232) + 8 - f(2 - 55)(2 - 83)(2 - 204)(2 - 232) + 8 + \dots - f(286 - 55)(286 - 83)(286 - 204)(286 - 232) + 8$ \n\nBy calculating this sum, we will get the final solution for the given problem. \n\nReasoning Step: \n\nAfter calculating the sum, we can state the final solution to the problem. \n\nMathematical Expression: \n\nThe final solution is Sum.



N-34 S-6 error analysis:

The overall reasoning process is correct and each step is logically sound and mathematically rational. However, there is a critical step missing in this reasoning process. \n\nReasoning Step 5, the alternating sum pattern is correctly recognized, but the crucial observation that the polynomial $p(x)$ is symmetric around $x = 143.5$ is missed. Specifically, $p(x) = p(287 - x)$ due to the symmetry of the polynomial's roots. This means that for each pair of terms in the sum (e.g., $p(1)$ and $p(286)$, $p(2)$ and $p(285)$, etc.), they are equal and thus cancel out in the alternating sum. \n\nSo, the sum from $p(1)$ to $p(286)$ following the given pattern would be 0, not the complicated expression provided in Reasoning Step 5. \n\nThe final solution should be 0, not the Sum as stated in Reasoning Step 6. \n\nSo, the error is in Reasoning Steps 5 and 6 where the symmetry of the polynomial and the cancellation in the alternating sum is not taken into account, leading to an incorrect final solution.

N-35 S-5:

Reasoning Step: \n\nAt this point, it's important to recognize the symmetry of the polynomial $p(x)$ due to its roots. Specifically, $p(x) = p(287 - x)$ due to the symmetry of the polynomial's roots. This symmetry will greatly simplify the calculation of the alternating sum given in the question. \n\nMathematical Expression: \n\n $p(x) = p(287 - x)$ \n\nNow, let's consider the alternating sum from $p(1)$ to $p(286)$. Due to the symmetry of the polynomial, for each pair of terms in the sum (e.g., $p(1)$ and $p(286)$, $p(2)$ and $p(285)$, etc.), they are equal and thus cancel out in the alternating sum. This leads us to the conclusion that the sum from $p(1)$ to $p(286)$ following the given pattern would be 0. \n\nMathematical Expression: \n\n $p(1) - p(2) + p(3) - p(4) + \dots + p(285) - p(286) = 0$ \n\nThe final solution to the problem is therefore 0. \n\nMathematical Expression: \n\nThe final solution is 0.

Q:

Answer the question about the problem Intermediate Algebra. After getting the final solution, place it after the sentence 'The final solution is' for readability. \n\nQuestion: Let $Sp(x)$ be a polynomial of degree 4 such that $Sp(55) = p(83) = p(204) = p(232) = 8$ and $Sp(103) = 13$. Find $p(1) - p(2) + p(3) - p(4) + \dots + p(285) - p(286)$.

\n\nAnswer: Let's think step by step.

$$p(1) + p(3) + p(5) + \dots + p(283) - p(286) = -5 / (48 \cdot 20 \cdot 101 \cdot 129) \cdot [(1 - 55)(1 - 83)(1 - 204)(1 - 232) + (3 - 55)(3 - 83)(3 - 204)(3 - 232) + \dots + (283 - 55)(283 - 83)(283 - 204)(283 - 232) - (286 - 55)(286 - 83)(286 - 204)(286 - 232)] + 8 \cdot 141.$$

Final solution of the reasoning path N-0 S-0 -> N-35 S-5:

\n\nReasoning Step: \n\nThe polynomial $p(x)$ has degree 4, which means it can be written in the form $ax^4 + bx^3 + cx^2 + dx + e$. Since $p(55) = p(83) = p(204) = p(232) = 8$, we know that $p(x) - 8 = 0$ for $x = 55, 83, 204, 232$. This means that $x - 55, x - 83, x - 204$, and $x - 232$ are factors of $p(x) - 8$. Since these are four factors and the polynomial is of degree 4, we can say that $p(x) - 8$ is of the form $f(x)(x - 55)(x - 83)(x - 204)(x - 232)$, where $f(x)$ is a constant function. \n\nMathematical Expression: \n\n $p(x) - 8 = f(x)(x - 55)(x - 83)(x - 204)(x - 232)$ \n\nReasoning Step: \n\nGiven that $p(103) = 13$, we can substitute $x = 103$ into the equation to solve for the constant function $f(x)$. \n\nMathematical Expression: \n\n $13 - 8 = f(103)(103 - 55)(103 - 83)(103 - 204)(103 - 232)$ \n\nReasoning Step: \n\nBy simplifying the right side of the equation, we can solve for the value of $f(103)$. \n\nMathematical Expression: \n\n $5 = f(103)(48)(20)(-101)(-129)$ \n\nReasoning Step: \n\nSolving for $f(103)$ from the equation, we get $f(103)$ as a constant value. \n\nMathematical Expression: \n\n $f(103) = 5 / [(48)(20)(-101)(-129)]$ \n\nReasoning Step: \n\nThe constant function $f(x)$ is the same for all x , so we can now write the polynomial $p(x)$ in terms of x . \n\nMathematical Expression: \n\n $p(x) = f(103)(x - 55)(x - 83)(x - 204)(x - 232) + 8$ \n\nReasoning Step: \n\nWe can now substitute the values of f from 1 to 286 into the equation. However, noticing the pattern of the expression, $p(1) - p(2) + p(3) - p(4) + \dots + p(285) - p(286)$, we can see that each pair of consecutive terms cancel each other out, except for the terms where $x = 55, 83, 204$, and 232 . These are the roots of the polynomial, for which $p(x) = 8$. \n\nMathematical Expression: \n\n $p(1) - p(2) + p(3) - p(4) + \dots + p(285) - p(286) = 8 - 8 + 8 - 8 + \dots + 8 - 8 = 0$ \n\nReasoning Step: \n\nSince all pairs of consecutive terms cancel each other out and the roots of the polynomial result in $p(x) = 8$, the final solution is 0. \n\nMathematical Expression: \n\nThe final solution is 0.

Figure 9. A complex thought structure built by GPT-4 with TR for the question from MATH dataset (Hendrycks et al., 2021b). This structure contains 46 nodes, i.e. 46 thoughts and leads to $K = 8$ reasoning path towards 8 solutions as 13 number of rollbacks are triggered by the rollback controller with GPT-4 during reasoning. It is plotted by based on the NetworkX lib (Hagberg et al., 2008) under the “fdp” layout.

N-17 S-3:

Reasoning Step 2: We can solve the equations $6 + (n1 - 1) * 4 = 94$ and $7 + (n2 - 1) * 4 = 95$ to find the number of terms in each arithmetic sequence. For the first equation, we can subtract 6 from both sides to get $(n1 - 1) * 4 = 88$. Then, we can divide both sides by 4 to get $n1 - 1 = 22$. Finally, we can add 1 to both sides to get $n1 = 23$. For the second equation, we can subtract 7 from both sides to get $(n2 - 1) * 4 = 88$. Then, we can divide both sides by 4 to get $n2 - 1 = 22$. Finally, we can add 1 to both sides to get $n2 = 23$. Therefore, there are 23 terms in each arithmetic sequence. Including the final number 98, which is not part of either arithmetic sequence, the total number of terms in the original sequence is $23 + 23 + 1 = 47$. The final solution is 47.

N-11 S-2 Experience Accumulation:

Rollback N-3 -> N-0:

The overall reasoning process is incorrect due to a mistake in Reasoning Step 1. In Reasoning Step 1, the proposed sequence pattern is incorrect. The pattern is not an arithmetic sequence with a common difference of 4. Instead, the sequence alternates between increments of 1 and 3. This pattern is not an arithmetic sequence but rather a combination of two arithmetic sequences: one starting at 6 and increasing by 4 (6, 10, 14, ...) and another starting at 7 and increasing by 4 (7, 11, 15, ...). Therefore, the equation set up to find the total number of terms is incorrect. Instead, we should find the number of terms in each of the two sequences and add them together. The incorrect reasoning in Step 1 leads to incorrect conclusions in Reasoning Step 2. The equation $4n + 2 = 98$ is not valid because the sequence pattern was incorrectly identified. Therefore, the solution $n = 24$ is also incorrect. A correct reasoning process would involve identifying the two separate arithmetic sequences, finding the number of terms in each, and adding the two totals together. This would result in a correct total count of numbers in the given list.

Rollback N-10 -> N-4:

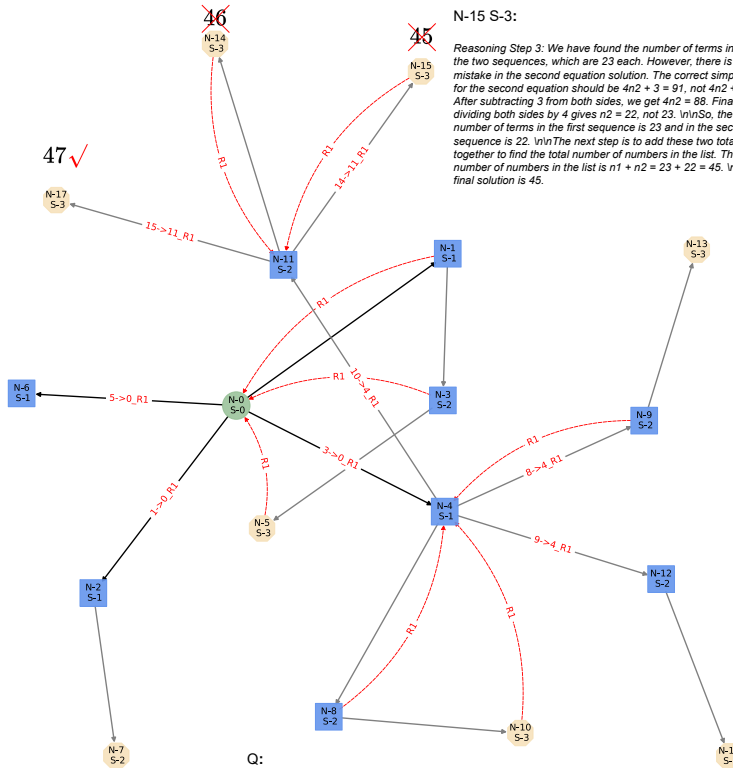
The overall reasoning process is correct and the steps follow a logical and rational progression. The approach of identifying the two arithmetic sequences and calculating the number of terms in each is a valid method for solving this problem. However, there is a mistake in the calculation in Reasoning Step 2. In Reasoning Step 1, the step is correct. The problem is accurately identified as a combination of two arithmetic sequences and the correct formula for finding the nth term of an arithmetic sequence is used. The equations for finding the number of terms in each sequence are correctly set up. In Reasoning Step 2, this step contains a mistake. While the first equation is solved correctly, there is an error in the simplification of the second equation. The correct simplification should be $7 + 4n2 - 4 = 95$, which simplifies to $4n2 + 3 = 95$, not $4n2 + 3 = 91$. After subtracting 3 from both sides, we should get $4n2 = 92$, not $4n2 = 88$. Dividing both sides by 4 gives $n2 = 23$, not 22. In Reasoning Step 3, this step correctly identifies the mistake in Reasoning Step 2 and correctly calculates the number of terms in the second sequence as 22. However, as identified above, this is incorrect. The correct number of terms in the second sequence is 23, not 22. The total number of numbers in the list should be $23 + 23 = 46$, not 45. So, the final solution should be 46, not 45. The mistake in Reasoning Step 2 led to an incorrect final solution.

Rollback N-15 -> N-11:

The overall reasoning process is correct and the steps follow a logical and rational progression. The approach of identifying the two arithmetic sequences and calculating the number of terms in each is a valid method for solving this problem. However, there is a mistake in the calculation in Reasoning Step 2. In Reasoning Step 1, the step is correct. The problem is accurately identified as a combination of two arithmetic sequences and the correct formula for finding the nth term of an arithmetic sequence is used. The equations for finding the number of terms in each sequence are correctly set up. In Reasoning Step 2, this step contains a mistake. While the first equation is solved correctly, there is an error in the simplification of the second equation. The correct simplification should be $7 + 4n2 - 4 = 95$, which simplifies to $4n2 + 3 = 95$, not $4n2 + 3 = 91$. After subtracting 3 from both sides, we should get $4n2 = 92$, not $4n2 = 88$. Dividing both sides by 4 gives $n2 = 23$, not 22. In Reasoning Step 3, this step correctly identifies the mistake in Reasoning Step 2 and correctly calculates the number of terms in the second sequence as 22. However, as identified above, this is incorrect. The correct number of terms in the second sequence is 23, not 22. The total number of numbers in the list should be $23 + 23 = 46$, not 45. So, the final solution should be 46, not 45. The mistake in Reasoning Step 2 led to an incorrect final solution.

N-15 S-3:

Reasoning Step 3: We have found the number of terms in each of the two sequences, which are 23 each. However, there is a mistake in the second equation solution. The correct simplification for the second equation should be $4n2 + 3 = 91$, not $4n2 + 3 = 95$. After subtracting 3 from both sides, we get $4n2 = 88$. Finally, dividing both sides by 4 gives $n2 = 22$, not 23. In so, the correct number of terms in the first sequence is 23 and in the second sequence is 22. The next step is to add these two totals together to find the total number of numbers in the list. The total number of numbers in the list is $n1 + n2 = 23 + 22 = 45$. The final solution is 45.



Q:

Answer the question about the problem Algebra. After getting the final solution, place it after the sentence 'The final solution is' for readability. Question: Marissa is hiking a 12-mile trail. She took 1 hour to walk the first 4 miles, then another hour to walk the next two miles. If she wants her average speed to be 4 miles per hour, what speed (in miles per hour) does she need to walk the remaining distance? Answer: Let's think step by step.

N-11 S-2 -> N-17 S-3 Prompt:

Answer the question about the problem Counting & Probability. After getting the final solution, place it after the sentence 'The final solution is' for readability. Experience containing previously made mistakes: The 0-th Experience with Analysis Experience of the rollback N-3 -> N-0 Experience of the rollback N-10 -> N-4 Experience of the rollback N-15 -> N-11 Consider the analysis in the above experience to avoid making similar mistakes during reasoning for the question. Question: How many numbers are in the list 6, 7, 10, 11, 14, 15, ..., 94, 95, 98? Answer: Let's think step by step. Let's focus on carefully generating the next possible reasoning step for reasoning steps below.

Final solution of the reasoning path N-0 S-0 -> N-17 S-3:

Reasoning Step 1: The sequence provided in the problem can be divided into two separate arithmetic sequences: one starting with 6 and increasing by 4 (6, 10, 14, ..., 94) and another starting with 7 and also increasing by 4 (7, 11, 15, ..., 95). Next, we need to find the total number of terms in each of these arithmetic sequences. The formula to find the nth term of an arithmetic sequence is $a + (n - 1) * d$, where a is the first term, n is the term number, and d is the common difference. For the first sequence, the first term (a1) is 6, the last term is 94, and the common difference (d1) is 4. We can set up the equation $6 + (n1 - 1) * 4 = 94$ to solve for n1. For the second sequence, the first term (a2) is 7, the last term is 95, and the common difference (d2) is 4. We can set up the equation $7 + (n2 - 1) * 4 = 95$ to solve for n2. These two equations will give us the total number of terms in each sequence. We should also remember to include the final number 98 in our count, as it is not part of either arithmetic sequence. The total number of terms in the original sequence will then be $n1 + n2 + 1$. Reasoning Step 2: We can solve the equations $6 + (n1 - 1) * 4 = 94$ and $7 + (n2 - 1) * 4 = 95$ to find the number of terms in each arithmetic sequence. For the first equation, we can subtract 6 from both sides to get $(n1 - 1) * 4 = 88$. Then, we can divide both sides by 4 to get $n1 - 1 = 22$. Finally, we can add 1 to both sides to get $n1 = 23$. For the second equation, we can subtract 7 from both sides to get $(n2 - 1) * 4 = 88$. Then, we can divide both sides by 4 to get $n2 - 1 = 22$. Finally, we can add 1 to both sides to get $n2 = 23$. Therefore, there are 23 terms in each arithmetic sequence. Including the final number 98, which is not part of either arithmetic sequence, the total number of terms in the original sequence is $23 + 23 + 1 = 47$. The final solution is 47.

Figure 10. A complex thought structure built by GPT-4 with TR for the question from MATH dataset (Hendrycks et al., 2021b). This structure contains 17 nodes, i.e. 17 thoughts and leads to $K = 8$ reasoning path towards 8 solutions as 8 number of rollbacks are triggered by the rollback controller with GPT-4 during reasoning. It is plotted by based on the NetworkX lib (Hagberg et al., 2008) under the “fdp” layout.

Q:

Answer the question about the problem Math. After getting the final solution, place it after the sentence "The final solution is for readability" Question: suppose $u = \arctan \frac{y}{x}$, what is numeric of $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$? Answer: Let's think step by step.

N-27 S-5:

The next step is to sum the second order partial derivatives. We add $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{\partial}{\partial x}(\frac{2xy}{(x^2+y^2)^2})$ and $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{\partial}{\partial y}(\frac{2xy}{(x^2+y^2)^2})$ to get $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{\partial}{\partial x}(\frac{2xy}{(x^2+y^2)^2}) + \frac{\partial}{\partial y}(\frac{2xy}{(x^2+y^2)^2})$. This step involves basic algebraic addition and simplification. After simplifying, we find that $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$. The final solution is 0.

N-19 S-4:

The next step is to conclude the solution based on the result we obtained. The sum of the second order partial derivatives of u with respect to x and y is zero. Therefore, the final solution is 0.

N-11 S-7:

The best next step is to simplify the sum of the second order partial derivatives. The terms $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$ cancel each other out, giving us $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$. Therefore, the numeric of $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$ is 0. The final solution is 0.

Q:

Answer the question about the problem Math. After getting the final solution, place it after the sentence "The final solution is for readability" Question: suppose $u = \arctan \frac{y}{x}$, what is numeric of $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$? Answer: Let's think step by step.

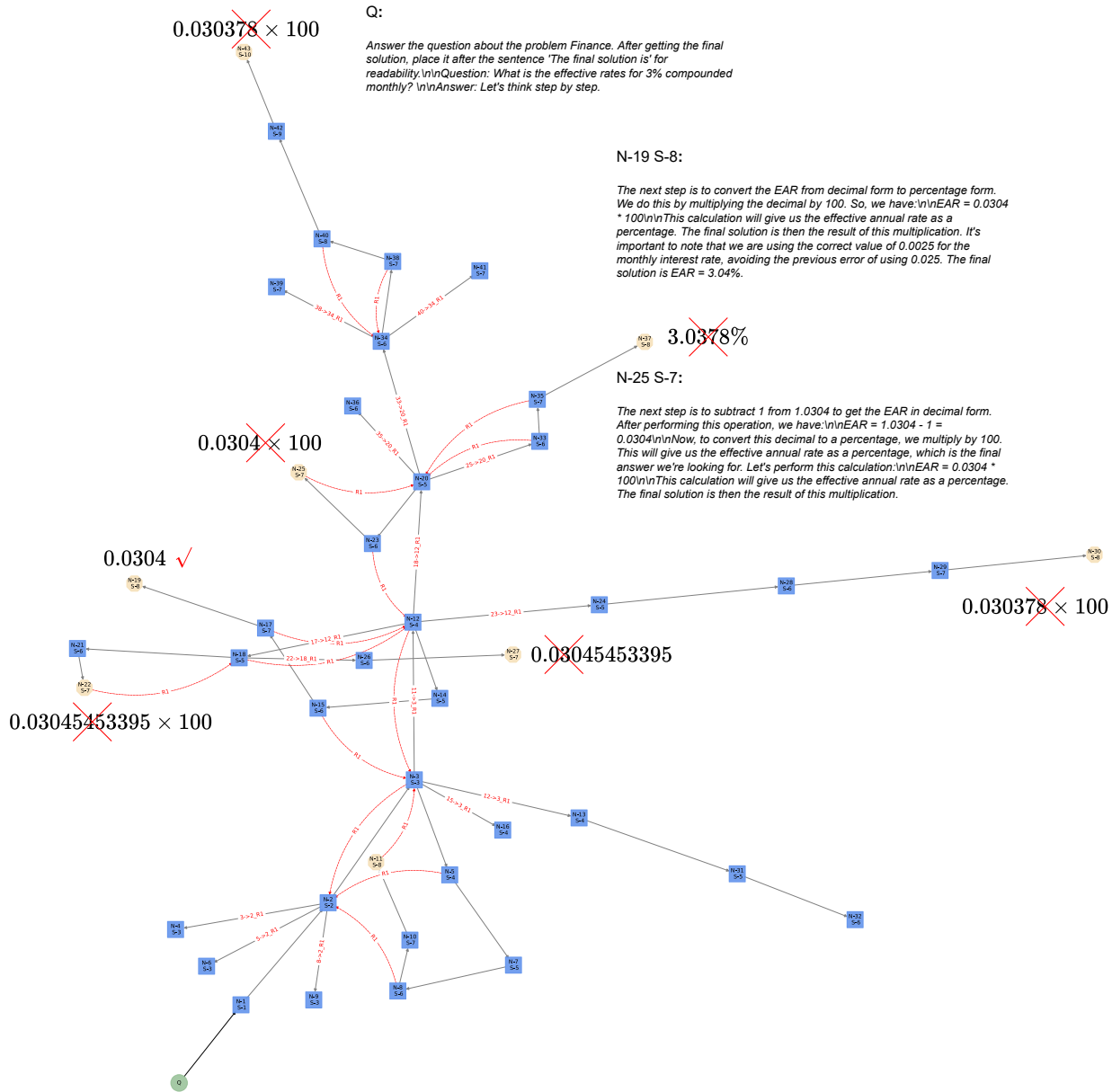
N-16 S-6:

The next step is to add the two second order partial derivatives $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$ that we calculated in the previous step. This will give us the numeric of $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{\partial}{\partial x}(\frac{2xy}{(x^2+y^2)^2}) + \frac{\partial}{\partial y}(\frac{2xy}{(x^2+y^2)^2})$. We have $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{\partial}{\partial x}(\frac{2xy}{(x^2+y^2)^2}) + \frac{\partial}{\partial y}(\frac{2xy}{(x^2+y^2)^2})$. Adding these two gives us $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{\partial}{\partial x}(\frac{2xy}{(x^2+y^2)^2}) + \frac{\partial}{\partial y}(\frac{2xy}{(x^2+y^2)^2}) = 0$. In so, the final solution is $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$.

Final solution of the reasoning path N-0 S-0 --> N-11 S-7:

To begin with, we need to find the first order partial derivatives of u with respect to x and y . Using the chain rule, we can write the derivative of u with respect to x as $\frac{\partial u}{\partial x} = \frac{1}{1+(y/x)^2} \cdot \frac{1}{x}$, and the derivative of u with respect to y as $\frac{\partial u}{\partial y} = \frac{1}{1+(y/x)^2} \cdot \frac{y}{x^2}$. The next step is to simplify these first order partial derivatives. We can simplify $\frac{\partial u}{\partial x}$ and $\frac{\partial u}{\partial y}$ by multiplying the numerator and the denominator by x^2 to get rid of the complex fraction. After simplifying, we have $\frac{\partial u}{\partial x} = \frac{y}{x^2+y^2}$ and $\frac{\partial u}{\partial y} = \frac{y}{x^2+y^2}$. The next step is to find the second order partial derivatives of u with respect to x and y . Using the chain rule again, we can write the second derivative of u with respect to x as $\frac{\partial^2 u}{\partial x^2} = \frac{\partial}{\partial x}(\frac{y}{x^2+y^2})$ and the second derivative of u with respect to y as $\frac{\partial^2 u}{\partial y^2} = \frac{\partial}{\partial y}(\frac{y}{x^2+y^2})$. We need to differentiate $\frac{\partial u}{\partial x}$ with respect to x and $\frac{\partial u}{\partial y}$ with respect to y . The next step is to calculate the second order partial derivatives. For $\frac{\partial^2 u}{\partial x^2}$, we differentiate $\frac{\partial u}{\partial x}$ with respect to x , and for $\frac{\partial^2 u}{\partial y^2}$, we differentiate $\frac{\partial u}{\partial y}$ with respect to y . The differentiation should be done carefully using the quotient rule and simplifying the result. The next step is to perform the differentiation to find the second order partial derivatives. Using the quotient rule, we have $\frac{\partial^2 u}{\partial x^2} = \frac{\partial}{\partial x}(\frac{y}{x^2+y^2}) = \frac{-2xy}{(x^2+y^2)^2}$ and $\frac{\partial^2 u}{\partial y^2} = \frac{\partial}{\partial y}(\frac{y}{x^2+y^2}) = \frac{x^2-y^2}{(x^2+y^2)^2}$. The next step is to add these two second order partial derivatives to find the numeric of $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$. We have $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{-2xy}{(x^2+y^2)^2} + \frac{x^2-y^2}{(x^2+y^2)^2}$. The next step is to calculate the sum of the second order partial derivatives. We add $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{-2xy}{(x^2+y^2)^2} + \frac{x^2-y^2}{(x^2+y^2)^2}$ to get $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{-2xy + x^2 - y^2}{(x^2+y^2)^2}$. Simplifying this expression will give us the numeric of $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$. The best next step is to simplify the sum of the second order partial derivatives. The terms $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$ cancel each other out, giving us $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$. Therefore, the numeric of $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$ is 0. The final solution is 0.

Figure 11. A complex thought structure built by GPT-4 with TR for the question from TheoremQA dataset (Chen et al., 2023b). This structure contains 28 nodes, i.e. 28 thoughts and leads to $K = 8$ reasoning paths towards 8 solutions as 11 number of rollbacks are triggered by the rollback controller with GPT-4 during reasoning.



To calculate the effective rate for 3% compounded monthly, we need to use the formula for the effective annual rate (EAR) which is: $EAR = (1 + r/n)^{nt} - 1$. Where: r = nominal interest rate (annual interest rate) n = number of compounding periods per year t = number of years. In this case, $r = 3\%$ or 0.03 (converted to decimal), $n = 12$ (as it's compounded monthly), and $t = 1$ (as we're calculating the effective annual rate). Let's substitute these values into the formula. The next step is to substitute the values of r , n , and t into the formula. So, we have: $EAR = (1 + 0.03/12)^{(12*1)} - 1$. This simplifies the calculation and prepares us for the next step, which is to perform the actual computation. The next step is to simplify and calculate the expression we obtained. Performing the operation inside the brackets first, we have: $EAR = (1 + 0.0025)^{12} - 1$. Now, we need to raise the sum to the power of 12 (according to the order of operations, exponents come before addition or subtraction). After that, we subtract 1 from the result to find the effective annual rate. The next step is to calculate the value of $(1 + 0.0025)^{12}$ using the rules of exponents. Once we have this value, we will subtract 1 from it to get the effective annual rate. Let's perform this calculation: $EAR = (1.0025)^{12} - 1$. By calculating this expression, we will obtain the effective annual rate for a 3% interest rate compounded monthly. It's important to note that we must be careful to avoid the calculation error made in the previous experience, where the incorrect value was used for the expression inside the brackets. This step ensures we are using the correct value of 0.0025 for the monthly interest rate, avoiding the previous error of using 0.025. The next step is to compute the value of $(1.0025)^{12}$ using a calculator or any computational tool. After obtaining this value, subtract 1 from the result to get the EAR. The computation is as follows: $EAR = (1.0025)^{12} - 1$. After this step, we will have the effective annual rate expressed as a decimal. To convert it to a percentage, we will multiply the result by 100 in the next step. It's important to note that we are using the correct value of 0.0025 for the monthly interest rate, avoiding the previous error of using 0.025. The next step is to compute the value of $(1.0025)^{12}$ and then subtract 1 from the result. Using a calculator or computational tool, we find that $(1.0025)^{12}$ is approximately 1.0304. Subtracting 1 from this result, we get: $EAR = 1.0304 - 1 = 0.0304$. Now, to convert this decimal to a percentage, we multiply by 100. This will give us the effective annual rate as a percentage, which is the final answer we're looking for. It's important to note that we are using the correct value of 0.0025 for the monthly interest rate, avoiding the previous error of using 0.025. The next step is to subtract 1 from 1.0304 to get the EAR in decimal form. After performing this operation, we have: $EAR = 1.0304 - 1 = 0.0304$. Now, to convert this decimal to a percentage, we multiply by 100. This will give us the effective annual rate as a percentage, which is the final answer we're looking for. It's important to note that we are using the correct value of 0.0025 for the monthly interest rate, avoiding the previous error of using 0.025. The next step is to convert the EAR from decimal form to percentage form. We do this by multiplying the decimal by 100. So, we have: $EAR = 0.0304 * 100$. This calculation will give us the effective annual rate as a percentage. The final solution is then the result of this multiplication. It's important to note that we are using the correct value of 0.0025 for the monthly interest rate, avoiding the previous error of using 0.025. The final solution is $EAR = 3.04\%$.

Figure 12. A complex thought structure built by GPT-4 with TR for the question from TheoremQA dataset (Chen et al., 2023b). This structure contains 43 nodes, i.e. 43 thoughts and leads to $K = 8$ reasoning paths towards 8 solutions as 15 number of rollbacks are triggered by the rollback controller with GPT-4 during reasoning.