Cognitive Loop of Thought: Reversible Hierarchical Markov Chain for Efficient Mathematical Reasoning

Anonymous ACL submission

Abstract

Multi-step Chain-of-Thought (CoT) has significantly advanced the mathematical reasoning capabilities of LLMs by leveraging explicit reasoning steps. However, the widespread adoption of Long CoT often results in sequence lengths that exceed manageable computational limits. While existing approaches attempt to alleviate this by reducing KV Cache redundancy via Markov chain-like structures, they introduce two critical limitations: inherent memorylessness (loss of context) and limited backward reasoning capability. To address these limitations, we propose a novel Chain-of-Thought framework based on Reversible Hierarchical Markov Chain, termed Cognitive Loop of Thought (CLoT), and a backward reasoning dataset CLoT-Instruct. In CLoT, problems are decomposed into sub-problems with hierarchical dependencies. Inspired by human cognitive processes—where reasoning is revisited to verify conclusions—we introduce a backward verification mechanism at each hierarchical layer. Furthermore, we implement a pruning strategy: once higher-level sub-problems are verified, redundant lower-level sub-problems are pruned to maximize efficiency. This approach effectively mitigates error propagation and enhances reasoning robustness. Experiments on four mathematical benchmarks demonstrate the effectiveness of our method. Notably, on the AddSub dataset using gpt-4o-mini, CLoT achieves 99.0% accuracy, outperforming traditional CoT and CoT-SC by 4.1% and 2.9%, respectively. Our code is publicly available at: https://anonymous.4open.science/r/CLoT-7EBD.

1 Introduction

Large Language Models (LLMs) have achieved remarkable breakthroughs across a broad spectrum of Natural Language Processing (NLP) tasks, such as question answering, automatic summarization, and machine translation (Achiam et al., 2023; Chowdhery et al., 2023; Touvron et al., 2023; Huang et al., 2023; Zhao et al., 2023). Despite these achievements, LLMs continue to exhibit notable limitations in reasoning performance when compared to human cognitive capabilities. Empirical evidence suggests that merely increasing model scale through parameter expansion is insufficient to bridge the gap in complex reasoning abilities between current LLMs and human-level intelligence. (Zhou et al., 2024).

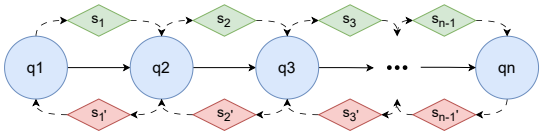


Figure 1: Overview of the Reversible Hierarchical Markov Chain Framework.

To enhance the model's reasoning capabilities, researchers have explored various strategies to enhance the logical coherence and step-by-step inference processes of LLMs. One of the most influential approaches is Chain-of-Thought (CoT) reasoning (Kojima et al., 2022), which enables models to generate intermediate reasoning steps before arriving at a final answer. By mimicking human-like problem-solving patterns, CoT not only improves model performance on complex reasoning tasks, but also enhances the interpretability of model decisions. The transparency afforded by explicit reasoning traces allows users to inspect, validate, and potentially correct the model's logic, thereby fostering trust and facilitating debugging.

However, when facing real-world application scenarios, CoT still encounters the challenge that intermediate reasoning steps may contain errors. To address this challenge, one approach to improve performance is intrinsic self-correction (Kamoi et al., 2024; Pan et al., 2024), which allows the model to check and revise its own generated answers without external feedback—a process highly analogous to human thinking. Nevertheless, neg-

ative views on self-correction also exist. (Stechly et al., 2023; Tyen et al., 2024; Jiang et al., 2024a) find that large language models are even unable to determine the correctness of answers, often changing correct answers into incorrect ones or failing to correct originally erroneous responses. The debates in prior research indicate that the self-correction mechanisms of large models still lack in-depth exploration.

To address these limitations and obviate the need for extensive training on verification tasks, we propose a novel reverse self-verification method named CLoT (Cognitive Loop of Thought). Humans often employ "reverse engineering" or self-validation to confirm inferred conclusions. Drawing inspiration from this cognitive process and reversible Markov chains, we treat the forward refinement and backward verification steps as a single closed loop, iteratively executing this process to ensure the validity of intermediate reasoning. Specifically, in the forward refinement phase, the LLM progressively generates sub-questions and answers via CoT. Subsequently, each sub-question undergoes backward verification: the original conditions are treated as unknown variables, while the generated answer serves as a known condition to deduce the original premises. Consistency between the backward inference and the original condition validates the step; discrepancies trigger a root-cause analysis. This bidirectional verification minimizes error propagation in the reasoning chain. We evaluate our method using gpt-4o-mini, gpt-4-1106-preview and DeepSeek-V3 across multiple mathematical, commonsense, and logical reasoning datasets. Our contributions are summarized as follows:

- We propose a novel self-reflective verification method for reasoning chains, named CLoT (Cognitive Loop of Thought). CLoT is capable of verifying both intermediate steps and the final answer, as well as analyzing the causes of errors.
- We introduce a hierarchical pruning strategy that removes redundant verification steps at lower layers, reducing CLoT's resource consumption by 41.8% without compromising performance.
- We construct CLoT-Instruct, an instruction-tuning dataset designed to facilitate the learning of "Backward Verification" capabilities in LLMs.
- Extensive experiments across six diverse tasks demonstrate that CLoT effectively identifies and corrects reasoning errors, consistently outperforming baseline methods.

2 Related Work

Since training LLMs requires enormous resources, effectively improving model performance without retraining has become a key research focus. (Wei et al., 2022) introduced the concept of CoT reasoning, emphasizing the importance of deriving conclusive answers through multi-step logical pathways. (Kojima et al., 2022) found that simply adding the phrase "let's think step by step" to prompts enables LLMs to perform zero-shot logical reasoning without any additional human-designed prompts. Subsequently, (Wang et al., 2023b) proposed Self-Consistency (SC) to replace the greedy decoding strategy. (Zhang et al., 2023) developed an automatic CoT framework based on the input problem, eliminating the instability caused by manual prompting. (Fu et al., 2023) employed complexity-based multi-step reasoning estimation to execute CoT. (Zhu et al., 2025) further enhanced performance by applying pre-prompting in the form of a plugin. However, standard chain-like reasoning still struggles with highly complex logical problems. Variants such as Tree-of-Thoughts (Yao et al., 2023), Graph-of-Thoughts (Besta et al., 2024), and Markov-chain-inspired reasoning frameworks have effectively enhanced the model's reasoning capabilities (Teng et al., 2025; Yang et al., 2025). Nevertheless, in practical applications, large models often fail to recognize their own errors and tend to repeat mistakes, limiting their applicability in real-world scenarios.

To expand the application scenarios of LLMs by enhancing their ability to identify and correct errors, researchers have developed various methods to improve their self-correction capabilities. FOBAR (Jiang et al., 2024b) relies on Self-Consistency, requiring multiple direct reasoning attempts (e.g., 10 times) for the same problem and triggering verification only when answers are inconsistent. (Wang et al., 2023a) verifies answers by integrating inputs from both humans and other models. (Weng et al.) employs backward verification to evaluate and score multiple candidate answers generated during a tree-of-thought reasoning process. (Li et al., 2024) enables models to effectively distinguish between confident and uncertain

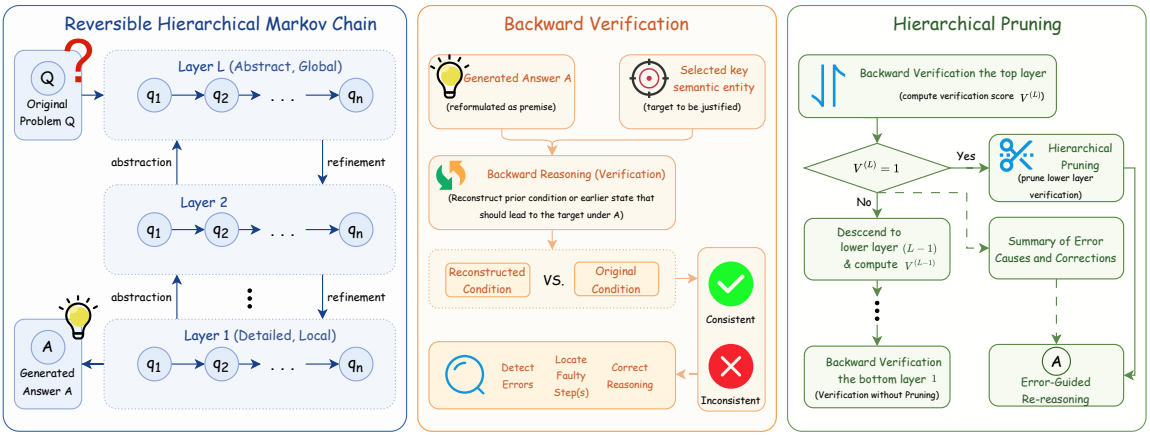


Figure 2: Overview of the CLoT Framework.

responses using a simple "Self-Verifying" prompt (e.g., "If you are very confident about your answer, maintain your answer. Otherwise, update your answer"). (Zheng et al., 2024) relies on ground truth to determine if a question needs to be answered again. (Chen and Li, 2024) identifies and corrects errors by having the LLMs analyze the intermediate steps of a CoT for mistakes and pinpoint the source of the error. While these methods have improved self-correction to varying degrees, they often face the challenges summarized by (Yang et al., 2024): (1) reliance on high-quality external feedback, (2) lack of comparison with baselines that consume the same number of tokens, and (3) prompts that are strictly designed and potentially complex.

To address these challenges, we propose the CLoT. Our approach achieves self-correction without requiring any external high-quality feedback. It demonstrates a significant performance improvement compared to a baseline with equivalent token consumption and operates in a zero-shot manner, eliminating the need for designing complex prompt examples.

3 Method

In this section, we provide a detailed description of the design of the CLoT method. CLoT guides the model to perform backward verification, identifies potential failures in backward verification, and ultimately summarizes the causes of errors and corrects them. Additionally, we construct a CLoT-Instruct Dataset based on the CLoT approach to help train models to learn "Backward Verification" approach.

3.1 Reversible Hierarchical Markov Chain

For a mathematical problem q, successful reasoning involves not only the sequential simplification of the problem through derivation steps but also an implicit hierarchical organization of cognitive processes—such as conceptual abstraction, subproblem decomposition, and strategic planning. To capture this structure, we propose a Hierarchical Markov Chain framework that models reasoning as a multi-level stochastic process, where each level corresponds to a distinct granularity of problem understanding or solution strategy.

The architecture of CLoT is illustrated in Figure 2, where the left panel delineates the structure of the Reversible Hierarchical Markov Chain. Specifically, CLoT first decomposes the target problem into a Markov chain of intermediate subproblems and explicitly outputs the logical dependencies among them. Subsequently, the model stratifies this sequence into discrete reasoning layers $l \in \{1, \ldots, L\}$ based on dependency counts. Layer L encompasses fewer dependencies and represents the most abstract strategy level, where the original problem is instantiated as the root node $q^{(L)}$. Conversely, layer 1 contains the most dependencies, representing the finest-grained derivation level. This cross-layer process of abstraction and refinement is probabilistically modeled as inter-layer propagation:

$$p(q^{(l+1)} \mid q^{(l)}) \quad \text{(abstraction)}, \tag{1}$$

$$p(q^{(l)} \mid q^{(l+1)}) \quad \text{(refinement)}. \tag{2}$$

Within each layer l, the reasoning process unfolds into an intra-layer Markov chain of subproblems $\{q_t^{(l)}\}_{t \geq 1}$, seeded by the layer's root repre-

sentation $q_1^{(l)}$. Given the current subproblem $q_t^{(l)}$ and its corresponding chain-of-thought reasoning step $s_t^{(l)}$ that resolves it, the transition to the next subproblem satisfies the Markov property:

$$p(q_{t+1}^{(l)} \mid q_t^{(l)}, s_t^{(l)}). \tag{3}$$

However, this hierarchical organization alone provides only a structural skeleton; it does not inherently guarantee the logical validity of each local derivation step. To address this and prevent error accumulation, we embed a reversible verification mechanism that operates layer by layer and subproblem by subproblem. To initiate this backward verification, the model-generated final answer is reformulated as an initial premise. The model then traverses the reasoning trajectory in reverse. At each step, rather than simply relying on forward generation, we introduce two complementary directional checks:

- **Forward Deduction**: The model derives the next state $q_{t+1}^{(l)}$ from $q_t^{(l)}$ via reasoning step $s_t^{(l)}$, representing standard local decision-making.
- **Backward Justification**: Given the consequent state $q_{t+1}^{(l)}$, the model verifies whether the antecedent state $q_t^{(l)}$ can be semantically and logically reconstructed. This serves as a semantic backtracking mechanism that asks: *Is the premise justifiable given the conclusion?*

A valid reasoning step must satisfy both directional validations. Through this step-by-step backward justification, the model ultimately attempts to reconstruct specific conditions from the original problem (or identify key semantic entities for non-numerical tasks). By determining whether these reconstructed target variables align with the original input, CLoT completes its dual-loop verification. This approach provides a fine-grained diagnostic signal to locate and rectify reasoning flaws or non sequiturs without requiring external supervision, parameter updates, or score-based candidate selection, thereby substantially enhancing the model's interpretability, safety, and reliability.

3.2 Hierarchical Pruning

Building upon the RHMC framework, we introduce a Consistency-Aware Hierarchical Pruning strategy. This method exploits hierarchical dependencies to minimize computational redundancy without sacrificing verification rigor. The core premise is that global semantic consistency at higher abstraction layers is a high-fidelity proxy for lower-level correctness. In CLoT's backward reconstruction, if all sub-questions within the highest layer L are successfully verified, the entire reasoning trajectory forms a closed logical loop. Formally, for a layer l with T_l sub-questions, we define the layer-wise verification status $\mathcal{V}^{(l)} \in \{0, 1\}$ as the logical conjunction of all individual sub-question outcomes:

$$\mathcal{V}^{(l)} = \prod_{t=1}^{T_l} \mathbb{I}\Big(\text{Verify}(q_t^{(l)} \mid q_{t+1}^{(l)})\Big), \tag{4}$$

where $\mathbb{I}(\cdot)$ is an indicator function returning 1 if the backward reconstruction of $q_t^{(l)}$ is strictly consistent, and 0 otherwise. This boolean gatekeeper drives a top-down dynamic pruning protocol:

- **Macro-Verification** & Pruning: The system evaluates from the top layer L. A perfect verification ($\mathcal{V}^{(l)} = 1$) indicates the final output is logically anchored to the input. Once confirmed, the system immediately prunes all lower-layer verifications, trusting that a reconstructible "big picture" implicitly validates fine-grained derivations.
- **Recursive Refinement**: If $\mathcal{V}^{(l)} = 0$, it signals a potential hallucination or logical gap. The system then descends to layer $l - 1$ for granular validation. This operates recursively: the model either halts and prunes upon achieving $\mathcal{V}^{(l)} = 1$ at any level, or descends until it pinpoints a specific failure at the finest-grained layer.

Crucially, this pruning is outcome-dependent. Subtle low-level errors that corrupt the final answer will inevitably cause high-level backward verification to fail. This automatically breaks the $\mathcal{V}^{(l)} = 1$ condition, forcing a mandatory descent. Thus, hierarchical pruning is triggered only when the reasoning is flawless across the entire semantic scope.

3.3 CLoT-Instruct Dataset

3.3.1 Dataset Construction

Building upon the theoretical framework of the RHMC, we introduce CLoT-Instruct, a specialized instruction-tuning dataset designed to train language models with bidirectional reasoning and self-verification capabilities. Unlike conventional

chain-of-thought (CoT) datasets that focus solely on forward reasoning, CLoT-Instruct explicitly encodes the backward verification mechanism central to our CLoT framework, enabling models to develop intrinsic self-consistency checking abilities.

Theoretically, our construction methodology is applicable to any mathematical reasoning dataset, as it does not rely on specific problem types but instead builds upon a general hierarchical reasoning structure and bidirectional verification logic. Currently, we have implemented and released three high-quality subsets based on this approach, using GSM8K, SVAMP, and AddSub as foundational datasets. The construction of CLoT-Instruct follows a two-stage pipeline designed to jointly ensure the quality of forward reasoning and the integrity of backward verification:

- **Forward Reasoning**: Creation of hierarchical problem-solving trajectories.
- **Backward Verification**: Generation of logically inverted validation questions.

3.3.2 Dataset Structure

Formally, let $\mathcal{D}_{\text{origin}}$ denote the original mathematical reasoning dataset, where each instance follows the format $\tau = (q, s_{1:T}, a)$. Here, q represents the initial problem, $s_{1:T}$ denotes a sequence of T reasoning steps ($T \geq 1$), and a is the final answer.

In contrast to conventional approaches that treat the entire reasoning chain as a monolithic unit, our CLoT framework explicitly decomposes reasoning into hierarchical components, each augmented with bidirectional verification signals. To support this, we construct the CLoT-Instruct dataset, in which every sample is uniformly structured as:

$$\mathcal{X} = \left(q_{\text{origin}},\ a_{\text{gt}},\ L,\ \{\tau^{(l)}\}_{l=1}^{L}\right), \tag{5}$$

where q_{origin} is the original problem, a_{gt} is its ground-truth answer, and L denotes the maximum reasoning depth required for this specific instance. For each reasoning layer $l \in \{1, \ldots, L\}$, the trajectory $\tau^{(l)}$ is defined as:

$$\tau^{(l)} = \left(q^{(l)},\ s_{1:T_l}^{(l)},\ \left\{(q_{\text{verify},t}^{(l)},\ a_{\text{verify},t}^{(l)})\right\}_{t=1}^{T_l}\right), \tag{6}$$

where $q^{(l)}$ is the problem representation at layer l (either the original question or an abstracted sub-question), and $s_{1:T_l}^{(l)}$ is its corresponding sequence of reasoning steps. Crucially, each pair $(q_{\text{verify},t}^{(l)}, a_{\text{verify},t}^{(l)})$ constitutes a bidirectional verification instance: $q_{\text{verify},t}^{(l)}$ is a logically inverted question derived from the forward step, and $a_{\text{verify},t}^{(l)}$ is its expected answer (representing the reconstructed antecedent).

This unified representation enables hierarchical dependency tracking, dynamic verification, and scalable supervision across various reasoning depths, all while maintaining a consistent and extensible data schema.

4 Experiment

In this section, we conduct comprehensive experiments to thoroughly investigate CLoT through extensive benchmark evaluations on six standard datasets, efficiency analysis, data result analysis, and ablation studies.

4.1 Experimental Settings

4.1.1 Datasets

We evaluate CLoT using gpt-4o-mini, gpt-4-1106-preview and DeepSeek-V3 as backbone models. Our evaluation covers three types of reasoning tasks: mathematical QA reasoning (MATH_{900}, (Hendrycks et al., 2021), GSM8K_{1319}, (Cobbe et al., 2021), Addsub_{395}, (Hosseini et al., 2014), and SVAMP_{1000}, (Patel et al., 2021)), mathematical multiple-choice (AQUA_{254}, (Ling et al., 2017)), and commonsense reasoning ($\text{CommonsenseQA}_{1220}$, (Talmor et al., 2019)). For the large datasets Math and SVAMP, we adopt the same data processing procedure as in Study (Chen and Li, 2024). On mathematical datasets, CLoT employs exact numeric matching—i.e., an answer is considered correct only if it exactly matches the numeric value in the original question. On commonsense datasets, it uses synonym-based judgment—i.e., an answer is deemed correct if it is a synonym or near-synonym of the answer in the original question.

4.2 Main Results

Table 1 summarizes the performance of CLoT and various baselines across six benchmarks using three different model backbones: gpt-4o-mini, gpt-4, and DeepSeek-V3. Overall, CLoT consistently demonstrates competitive performance, outperforming all competitive baselines in terms of average accuracy.

Under the gpt-4o-mini setting, CLoT achieves an average accuracy of 89.6%, surpassing the most competitive baseline, CoT-SC (n=5), by a margin of

Models	AddSub	GSM8K	SVAMP	MATH	AQuA	CSQA	Avg.
gpt-4o-mini							
CoT	94.9	90.9	93.4	78.3	81.4	81.5	86.7
CoT-SC(n=5)	96.1	92.0	93.5	81.0	83.1	83.4	88.3
ToT	76.7	91.5	93.2	80.7	72.3	81.7	82.7
AR	82.3	91.7	92.4	73.1	77.5	78.7	81.0
C-CoT	97.2	92.4	93.4	76.9	83.3	81.0	87.4
ISP-2	96.1	92.4	93.9	77.9	83.7	81.9	87.7
CLoT(ours)	99.0±0.0	94.6±0.2	94.9±0.1	80.7±0.3	85.8±0.0	82.3±0.3	89.6
gpt-4							
CoT	98.0	93.1	93.4	73.0	82.7	84.9	87.5
CoT-SC(n=5)	98.4	94.2	93.5	81.6	83.1	85.6	89.4
AR	87.1	93.4	92.7	73.1	79.5	79.1	84.2
C-CoT	98.4	94.2	93.6	80.6	82.7	83.9	88.9
ISP-2	97.7	93.3	93.5	77.7	84.2	83.4	88.3
Tr	97.5	94.2	91.3	71.9	79.9	79.4	85.7
CLoT(ours)	99.0±0.0	95.4±0.4	95.0±0.1	81.6±0.6	86.9±0.0	84.9±0.1	90.5
DeepSeek-V3							
CoT	97.7	94.1	93.9	90.2	84.2	81.1	90.2
CoT-SC(n=5)	98.5	94.5	94.3	91.3	87.4	82.2	91.4
AR	93.9	93.3	93.1	89.7	86.6	81.0	89.6
C-CoT	97.9	94.2	93.8	89.7	88.1	81.2	90.5
CLoT(ours)	99.0±0.0	95.5±0.2	95.6±0.2	91.7±0.2	91.8±0.3	82.6±0.8	92.7

Table 1: Accuracy of CLoT and baselines on six mainstream datasets. Higher is better for all metrics. The best score, and second best score are red, and orange, respectively.

1.3%. Notably, CLoT establishes new performance ceilings on AddSub (99.0%), GSM8K (94.6%), SVAMP (94.9%), and AQuA (85.8%), demonstrating its robust reasoning capabilities across both straightforward arithmetic and complex multi-step reasoning tasks. When evaluated on the more powerful gpt-4 backbone, CLoT further extends its lead, attaining the highest average accuracy of 90.5%. Compared to the best-performing baseline, it delivers substantial gains on GSM8K (+1.2%), SVAMP (+1.5%), and AQuA (+3.8%). These results underscore CLoT's ability to effectively leverage its iterative generate-and-verify mechanism to rectify subtle errors in the reasoning chain, particularly in challenging datasets like AQuA. Furthermore, CLoT demonstrates superior scalability and generalization on DeepSeek-V3, achieving a peak average accuracy of 92.7%. The consistent improvements across diverse tasks, ranging from symbolic arithmetic (AddSub) to complex mathematical reasoning (MATH) and CSQA, validate the effectiveness and versatility of our proposed verification driven framework.

4.3 Efficiency Analysis

We further evaluate the efficiency of different methods in terms of token consumption. The results are shown in Figure 3 and were obtained using

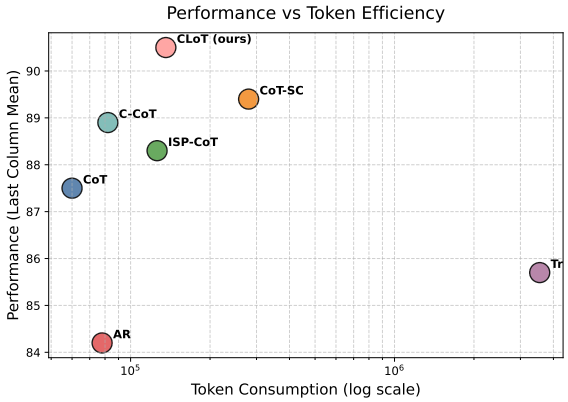


Figure 3: The results of CLoT efficiency. The token consumption corresponds to the total number of tokens used to solve the same 100 gsm8k problem.

the gpt-4o-mini API. Traditional methods such as CCoT and CoT-SC consume approximately 280k tokens. In contrast, Tr uses chain-of-thought for self-verification and requires as many as 3.3M tokens, resulting in very high computational costs. AR is more token-efficient, using only 78k tokens. However, its overall performance still lags behind other approaches. Notably, CLoT achieves performance on par with or even better than existing methods while consuming just 136k tokens. This highlights its strong reasoning efficiency under low

token overhead. This efficiency stems from its hierarchical pruning strategy, which selectively removes redundant verification steps at lower reasoning layers. This design enables high reasoning efficacy with minimal token overhead, making CLoT particularly suitable for large-scale and resource-constrained applications.

4.4 Verification Analysis

To systematically evaluate our model's verification capability on mathematical reasoning, we built verification datasets from four standard math QA benchmarks: AddSub, SVAMP, GSM8K, and AQuA (see Figure 4). Error samples are identified through strict comparison between generated final answers and Ground Truth (GT), with any discrepancy flagged as incorrect. Second, backward verification operates exclusively on the model's own generated reasoning trajectory, ensuring self-correction is fully grounded in internal context. Finally, the maximum layer count is dynamically controlled by the problem's maximum dependency number. We analyze three key metrics:

- One-Step Verification Accuracy: the fraction of problems whose initial solution is both correct and verified on the first attempt.
- Total Verification Accuracy: the proportion of problems that are either correctly solved or whose errors are successfully detected.
- Error Omission Rate: the fraction of incorrect solutions missed by the verifier.

One-Step Verification Accuracy is high on AddSub (97.8%) and GSM8K (89.4%), indicating strong self-verification on problems with simple or clear reasoning paths. It drops sharply on SVAMP (30.8%) and AQuA (13.8%), reflecting the difficulty for models to produce completely flawless and verifiable solutions in a single generation for semantically complex or multi-step reasoning problems. In contrast, Total Verification Accuracy remains consistently high across all datasets, demonstrating that CLoT reliably ensures output correctness by either producing correct answers or flagging errors. Correspondingly, the Error Omission Rate is low, confirming the verifier's high recall. Notably, AQuA achieves an omission rate of only 2.8%, comparable to GSM8K, despite its low one-step accuracy. This highlights the effectiveness of multi-round verification in catching errors even from poor initial solutions. Finally, the maximum layer counts are shown in Figure 4b. The results indicate that AddSub and SVAMP have at most two layers of subproblems, whereas GSM8K and AQuA exhibit three or more layers, demonstrating that more challenging tasks require multiple rounds of the "generate-and-verify" cycle.

Model	Accuracy
Qwen-2.5-1.5B	75.33%
Qwen-2.5-1.5B-CLoT-Instruct	81.67%

Table 2: SFT Performance Comparison

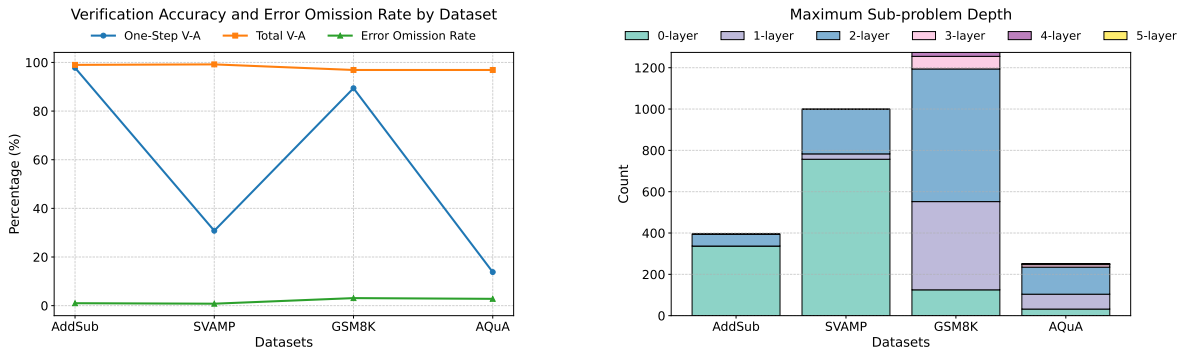
4.5 Datasets Analysis

CLoT-Instruct utilizes instruction tuning to enhance models' self-verification capabilities. We fine-tuned Qwen-2.5-1.5B on CLoT-GSM8K-Instruct (2,700 training, 300 test samples). To mitigate trajectory inconsistency, we evaluated performance strictly via final answer matching, the results are shown in Table 2. The fine-tuned model's GSM8K accuracy rose from 75.33% to 81.67% (+6.34%), proving our data quality and method's efficacy. Future work will scale this pipeline to larger datasets and explore automated trajectory filtering to boost generalization and robustness in complex reasoning.

4.6 Ablation Analysis

To systematically evaluate the effectiveness of each component in our proposed method, we conduct a series of ablation studies using the gpt-4o-mini model. As shown in Table 3, we incrementally introduce four key designs: (1) standard Chain-of-Thought (CoT) with a single direct answer verification (CoT + Self-Verifying, or CoT + SV), serving as the baseline; (2) CoT enhanced with Hierarchical Markov Chain decomposition (CoT + HMC) without any verification, to assess the benefit of structured reasoning alone; (3) CoT with Reversible Hierarchical Markov Chain (CoT + RHMC), which adds step-by-step verification at each reasoning level to improve overall reliability; and (4) CoT + RHMC further augmented with Hierarchical Pruning (HP), where lower-level verification steps are skipped once all subproblems at a higher level are confirmed correct.

The results indicate that CoT+SV yields only a marginal improvement over the baseline CoT. This highlights the limitations of unstructured ver-



(a) (b)

Figure 4: We present a visualization of the dataset analysis. Figure (a) shows the various accuracy metrics on the dataset, where V-A denotes Verification Accuracy. Figure (b) shows the maximum number of layers in the multi-layer Markov chains used in the dataset.

Models	Token	AddSub	GSM8K	SVAMP	AQuA	Avg.
CoT (baseline)	60k	94.9	90.9	93.4	81.4	90.2
CoT + SV	98k	94.0	93.5	92.3	81.4	90.3
CoT + HMC	112k	94.9	90.9	93.4	81.0	90.1
CoT + RHMC	325k	94.4	93.5	93.9	84.6	91.6
CoT + RHMC + HP	136k	99.0	94.6	94.9	85.8	93.6

Table 3: Ablation study results of CLoT based on gpt-4o-mini. The experiment used four datasets and token consumption to evaluate the contributions of three components. The token consumption corresponds to the total number of tokens used to solve the same 100 GSM8K problems (idx0 – idx99).

ification, which often struggles with complex error localization. When upgrading to CoT+RHMC, the average accuracy significantly improves to 91.6%. However, this method generates excessively long reasoning chains (325k tokens), requiring the model to meticulously verify every low-level sub-problem even when the high-level logic is already sound. Finally, CoT+RHMC+HP further boosts the average accuracy to 93.6% while drastically reducing token consumption to 136k. Although pruning is fundamentally an efficiency mechanism, its accuracy gains stem from the attention characteristics of LLMs. Excessively long reasoning chains cause attention dilution, whereas HP maintains concise reasoning chains by pruning redundant checks after macroscopic consistency is verified, thereby preventing the model from over-correcting valid reasoning steps.

5 Conclusion

In this paper, we introduced Cognitive Loop of Thought (CLoT), a novel reasoning framework that addresses the inherent "memorylessness" and error propagation issues in traditional Chain-of-Thought (CoT) prompting. By modeling the reasoning process as a Reversible Hierarchical Markov Chain (RHMC), we bridge the gap between sequential token generation and human-like cognitive verification. Our core contribution, the backward verification mechanism, enables LLMs to treat reasoning as a closed loop, where conclusions are validated by reconstructing original premises. To ensure this process remains computationally viable, we implemented a hierarchical pruning strategy that leverages the dependency structure of sub-problems to bypass redundant checks. Our experiments across six benchmarks demonstrate that CLoT consistently achieves state-of-the-art performance, notably reaching 99.0% accuracy on the AddSub dataset. Furthermore, our ablation studies confirm that the synergy between reversible logic and hierarchical pruning allows for a 41.8% reduction in token overhead without sacrificing reasoning integrity. The release of CLoT-Instruct provides a foundational resource for the community to further explore bidirectional instruction tuning. We believe that shifting from purely forward-moving chains to cognitive loops is a critical step toward developing more reliable, self-correcting, and autonomous reasoning agents.

Limitations

This study has two main limitations. First, CLoT relies on the model's intrinsic ability to perform backward reasoning. Our experiments show that while large-scale models like gpt-4o excel at this, smaller or less capable models may struggle with the logical inversion required for effective self-verification, potentially leading to false negatives during the pruning process. Second, our current evaluation focuses primarily on mathematical, logical, and commonsense reasoning tasks where conclusions are relatively deterministic. In more subjective or open-ended tasks, such as creative writing or legal argumentation, defining a clear backward verification logic is more challenging, as there may not be a unique set of premises that lead to a specific conclusion.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, and et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Maciej Besta, Nils Blach, Ales Kubicek, and et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 17682–17690.

Sijia Chen and Baochun Li. 2024. Toward adaptive reasoning in large language models with thought rollback. In *Proceedings of the 41st International Conference on Machine Learning*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, and et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, and et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Yao Fu, Hao Peng, Ashish Sabharwal, and et al. 2023. Complexity-based prompting for multi-step reasoning. In *The Eleventh International Conference on Learning Representations*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, and et al. 2021. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 523–533.

Jiaxin Huang, Shixiang Gu, Le Hou, and et al. 2023. Large language models can self-improve. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1051–1068.

Dongwei Jiang, Jingyu Zhang, Orion Weller, and et al. 2024a. Self-[in] correct: Llms struggle with refining self-generated responses. *CoRR*.

Weisen Jiang, Han Shi, Longhui Yu, and et al. 2024b. Forward-backward reasoning in large language models for mathematical verification. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6647–6661, Bangkok, Thailand. Association for Computational Linguistics.

Ryo Kamoi, Yusen Zhang, Nan Zhang, and et al. 2024. When can llms actually correct their own mistakes? a critical survey of self-correction of llms. *Transactions of the Association for Computational Linguistics*, 12:1417–1440.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, and et al. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213.

Loka Li, Zhenhao Chen, Guangyi Chen, and et al. 2024. Confidence matters: Revisiting intrinsic self-correction capabilities of large language models. *arXiv preprint arXiv:2402.12563*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167.

Liangming Pan, Michael Saxon, Wenda Xu, and et al. 2024. Automatically correcting large language models: Surveying the landscape of diverse automated correction strategies. *Transactions of the Association for Computational Linguistics*, 12:484–506.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094.

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

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158.

Fengwei Teng, Zhaoyang Yu, Quan Shi, and et al. 2025. Atom of thoughts for markov llm test-time scaling. *arXiv preprint arXiv:2502.12018*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, and et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Gladys Tyen, Hassan Mansoor, Victor Carbune, and et al. 2024. LLMs cannot find reasoning errors, but can correct them given the error location. pages 13894–13908. Association for Computational Linguistics.

Tianlu Wang, Ping Yu, and et al. 2023a. Shepherd: A critic for language model generation. *arXiv preprint arXiv:2308.04592*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, and et al. 2023b. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, and et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837.

Yixuan Weng, Minjun Zhu, Fei Xia, and et al. Large language models are better reasoners with self-verification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2550–2575.

Wen Yang, Minpeng Liao, and Kai Fan. 2025. Markov chain of thought for efficient mathematical reasoning. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7132–7157.

Zhe Yang, Yichang Zhang, Yudong Wang, and et al. 2024. Confidence vs critique: A decomposition of self-correction capability for llms. *arXiv preprint arXiv:2412.19513*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, and et al. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822.

Michihiro Yasunaga, Xinyun Chen, Yujia Li, and et al. 2024. Large language models as analogical reasoners. In *The Twelfth International Conference on Learning Representations*.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, and et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2).

Chuanyang Zheng, Zhengying Liu, Enze Xie, and et al. 2024. Progressive-hint prompting improves reasoning in large language models. In *AI for Math Workshop @ ICML 2024*.

Lexin Zhou, Wout Schellaert, Fernando Martínez-Plumed, and et al. 2024. Larger and more instructable language models become less reliable. *Nature*, 634(8032):61–68.

Dong-Hai Zhu, Yu-Jie Xiong, Jia-Chen Zhang, and et al. 2025. Understanding before reasoning: Enhancing chain-of-thought with iterative summarization pre-prompting. *arXiv preprint arXiv:2501.04341*.

A datasets

A.1 Mathematical Reasoning Datasets

We evaluate CLoT on four mathematical QA datasets and one multiple-choice dataset:

1) MATH (Hendrycks et al., 2021)& SVAMP (Patel et al., 2021): Given the extensive size of these benchmarks, we follow the established data processing procedure from prior research (Chen and Li, 2024), specifically evaluating the first 900 samples of MATH and the first 1,000 samples of SVAMP to ensure consistent comparison.

2) GSM8K (Cobbe et al., 2021)& AddSub (Hosseini et al., 2014): We utilize the full test sets (1,319 and 395 samples, respectively) to assess the model's ability to handle grade-school level multi-step arithmetic word problems.

3) AQuA (Ling et al., 2017): A mathematical multiple-choice dataset requiring algebraic reasoning. This helps evaluate CLoT's performance when the solution space is constrained by predefined options.

A.2 Commonsense Reasoning Dataset

CommonsenseQA (CSQA) (Talmor et al., 2019): We use 1,220 samples from CSQA to test the framework's ability to leverage world knowledge and semantic relationships.

Dataset	Reasoning Task	Answer Type	Number
MATH	Mathematical	Number	900
GSM8K	Mathematical	Number	1,319
AddSub	Mathematical	Number	395
SVAMP	Mathematical	Number	1,000
AQuA	Mathematical	Multi-choice	254
CSQA	Commonsense	Multi-choice	1,220

Table 4: Overview of datasets utilized in CLoT experiments.

A.3 License

All datasets used in this study are publicly available resources employed strictly for non-commercial research purposes. MATH (Hendrycks et al., 2021) and SVAMP (Patel et al., 2021) are both released under the MIT License. GSM8K (Cobbe et al., 2021) is made available by OpenAI under the MIT License, while AQuA (Ling et al., 2017) is distributed under the Apache License 2.0. AddSub (Hosseini et al., 2014) and CommonsenseQA (CSQA) (Talmor et al., 2019) do not specify explicit open-source licenses; both are publicly accessible and widely adopted for academic research in the NLP community.

B Prompt Cases

To support reproducible research, the full prompt templates utilized in our experimental pipeline are publicly released.

Initial Reasoning

System Prompt:
You are a precise math problem solver. Solve the given math problem step by step:

Please convert fractional answers to decimal form.
Please ensure that a final numerical answer is obtained.
Please extend your chain of thought as much as possible; the longer the chain of thought, the better.

You can freely reason in your response, but please enclose the final answer within <answer></answer> tags (pure number without units and explanations)

Question:(...)

Response:
<think>(...)</think>
<answer>(...)</answer>

Problem Decomposition

System Prompt:
You are tasked with breaking down a math problem reasoning process into sub-questions. Please ensure that there are no repeated questions.

Instructions:
1. Break down the reasoning process into a series of sub-questions
2. Each sub-question should:
- Do not ask for equation
- Be written in interrogative form
- Have a clear numerical answer
- List its other sub-questions' indexes it depends (0-based, can be an empty list)

Task	Answer-format Instructions
GSM8K, SVAMP, AddSub, math	You can freely reason in your response, but please enclose the final answer within <answer></answer>tags(pure number without units and explanations)
AQuA, CommonsenseQA	You can freely reason in your response, but please enclose the final option within <answer></answer>tags(pure uppercase option without explanations)

Table 5: Answer-format instructions for different tasks.

3. Dependencies are defined as information needed to answer the current sub-question that:
- Does NOT come directly from the original question
- MUST come from the answers of previous sub-questions

formatter = """
Format your response as the following JSON object:
{{
"sub-questions": [
{{
"description": "<clear interrogative question>",
"answer": <numerical value without units>,
"depend": [<indices of prerequisite sub-questions>]
}},
...
],
"answer": answer
}}
Question:(...)
Chain of Thought:(...)

Response:
<think>(...)</think>
{{
"sub-questions": [
{{
"description": "(...)",
"answer": (...),
"depend": [(...)]
}},
...
],
"answer": answer
}}

Problem Reconstruction

System Prompt:
Please replace one numerical value in the original problem with x. This value will be used to construct a verification question that checks the correctness of the original problem.
Only one numerical value should be replaced with x; all other information must remain unchanged. Whenever possible, avoid using fractions or decimals when introducing x.
Please enclose the modified question within <question></question> tags and enclose The original value that was replaced by x (only one) within <answer></answer> tags (pure number without units and explanations)

Question:(...)

Response:
<question>(...)</question>
<answer>(...)</answer>

Supplementary Information

System Prompt:
Please add the necessary information to the sub-question so that it can be answered independently of the original question.

Instructions:
1. Search for information in the original question that is relevant to solving the sub-question.
2. Combine the information with the sub-

question in a logical way to form a new question that can be solved independently.
3. If the sub-question is consistent with the original question, directly output the original question as the new question.

You are not required to solve the problem, but please enclose the new problem within <problem></problem> tags.

Question:(...)

Response:
<problem>(...)</problem>

Conclusion

System Prompt:
You are given a main question and a related sub-question. Your task is to analyze how the sub-question contributes to solving the main question, and to summarize the key strategies or patterns used in correctly answering the sub-question. Focus on identifying problem-solving techniques and any dependencies between the sub-question and the final solution.

Please enclose the problem-solving techniques within <solve></solve> tags.

Question:(...)
Sub-question:(...)

Response:
<solve>(...)</solve>

Final Reasoning

System Prompt:
You are a precise math problem solver. Solve the given math problem step by step. Please refer to the following sub-questions, sub-answers and problem-solving experiences.

Please convert fractional answers to decimal form.
Please ensure that a final numerical answer is obtained.

You can freely reason in your response, but please enclose the final answer within <answer></answer> tags (pure number without units and explanations)

Question:(...)
Sub-question:(...)

Response:
<think>(...)</think>
<answer>(...)</answer>

C Answer Formation

The following section illustrates the multi-stage reasoning process of CLoT, where discrete steps are facilitated through Chain-of-Thought (CoT) prompting. Specifically, Tables 6 and 8 exemplify the detailed reasoning trajectories for mathematical problems. To ensure consistency and facilitate evaluation, the final output generation phase adheres to the standardized formats presented in Table 5.

SVAMP Problem
Q: Kylar went to the store to buy glasses for his new apartment. One glass costs $5, but every second glass costs only 60% of the price. Kylar wants to buy 16 glasses. How much does he need to pay?
Step 1, First CoT
Q: Let's think step by step. ***Thought*** <answer> 64 </answer>
Step 2, Backward Verification
Q: Please replace one numerical value in the original problem with X and incorporate the answer as a known condition. Kylar went to the store to buy glasses for his new apartment. One glass costs X, but every second glass costs only 60% of the price. Kylar wants to buy 16 glasses. Knowing that he need to pay $64 for them. What's the X?
Step 3, Solve the new problem
Q: Let's think step by step. ***Thought*** <answer> 5 </answer>
Step 4, Judge
$5 = 5$
Answer: 64

Table 6: Examples of Successful Verification by CLoT on Mathematical Problem Datasets.

Method	Accuracy (%)
CoT	26.67
CoT-SC	23.33
CLoT (Ours)	30.00

Table 7: Performance comparison on the AIME dataset (gpt-4).

D AIME Result

We have supplemented our experiments with the AIME using gpt-4. The results, summarized in Table 7, show that CLoT (30.00%) outperforms both CoT (26.67%) and CoT-SC (23.33%). Notably, on high-difficulty tasks, the performance of CoT-SC degrades, whereas CLoT continues to improve. This demonstrates that our verification mechanism is more effective than simple self-consistency sampling in complex reasoning scenarios.

E Declaration of AI Assistance

In the preparation of this manuscript, AI-assisted tools were used exclusively for language polishing, grammar correction, and stylistic refinement of the text. These tools were not employed in the conception or design of the research methodology, the development of algorithms, the implementation of code, the analysis of experimental results, or the formulation of core scientific claims.

F Algorithm

This section formalizes the operational workflow of the **Cognitive Loop of Thought** (CLoT) framework in Algorithm 1. CLoT achieves robust multi-step reasoning through hierarchical state transitions that decompose problems into layered Markov chains, coupled with a reversible verification mechanism that validates logical validity via backward justification and eliminates redundant computation through consistency-aware hierarchical pruning. These mechanisms are theoretically grounded in Reversible Hierarchical Markov Chains (RHMC).

Algorithm 1: CLoT Reasoning

Input: Original problem $q_1^{(L)}$ at top layer L
Output: Final answer a and global verification status $\mathcal{V}$

```
// Forward: top-down hierarchical deduction
for l = L to 1 do
    t ← 1;
    while not terminated do
        q_{t+1}^{(l)} ~ p(q_{t+1}^{(l)} | q_t^{(l)}, s_t^{(l)});
        t ← t + 1;
    end
    T_l ← t;
    if l > 1 then q_1^{(l-1)} ~ p(q_1^{(l-1)} | q_{1:T_l}^{(l)});
end
a ← q_{T_1}^{(1)};
// Backward: top-down verification and pruning
for l = L to 1 do
    V ← 1;
    for t = T_l − 1 to 1 do
        if Verify(q_t^{(l)} | q_{t+1}^{(l)}) fails then
            V ← 0; break;
    end
    if V = 1 then break ;
     // Hierarchical pruning
    ;
end
return a, V
```

G Baseline

Our baselines include classical prompting methods (CoT, (Kojima et al., 2022)), CoT with Self-Consistency (CoT-SC, n = 5, (Wang et al., 2023b)), CoT with Iterative Summarization Pre-Prompting (ISP-2, (Zhu et al., 2025)), Analogical Reasoning (AR, (Yasunaga et al., 2024)), Complex-CoT (C-CoT, (Fu et al., 2023)) and Thought Rollback (Tr, (Chen and Li, 2024)).

SVAMP Problem
Q: Kylar went to the store to buy glasses for his new apartment. One glass costs $5, but every second glass costs only 60% of the price. Kylar wants to buy 16 glasses. How much does he need to pay?
Step 1, First CoT
Q: Let's think step by step. ***Thought*** <answer> 96 </answer>
Step 2, Backward Verification
Q: Please replace one numerical value in the original problem with X and incorporate the answer as a known condition. Kylar went to the store to buy glasses for his new apartment. One glass costs X, but every second glass costs only 60% of the price. Kylar wants to buy 16 glasses. Knowing that he need to pay $64 for them. What's the X?
Step 3, Solve the new problem
Q: Let's think step by step. ***Thought*** <answer> 10 </answer>
Step 4, Judge
$4 \neq 5$
Step 5, Decompose sub-problems
Q: You are tasked with breaking down a math problem reasoning process into sub-problems. q1, q2, q3,...
Step 6, For each subproblem, iteratively execute Steps 1–4.
If all subproblems at the same level pass verification, exit the loop.
Step 7, Second CoT
Q: Let's think step by step. Please refer to the following sub-problems and sub-answers. ***Thought*** <answer> 64 </answer>
Answer: 64

Table 8: Examples of CLoT's verification failures on mathematical problem datasets.