Atom of Thoughts for Markov LLM Test-Time Scaling

Anonymous ACL submission

Abstract

Large Language Models (LLMs) achieve superior performance through training-time scaling, and test-time scaling further enhances their capabilities by conducting effective reasoning during inference. However, as the scale of reasoning increases, existing test-time scaling methods suffer from accumulated historical information, which not only wastes computational resources but also interferes with effective reasoning. To address this issue, we observe that complex reasoning progress is often achieved by solving a sequence of independent subquestions, each being self-contained and verifiable. These subquestions are essentially *atomic questions*, relying primarily on their current state rather than accumulated history, similar to the memoryless transitions in a Markov process. Based on this observation, we propose Atom of Thoughts (**AoT**), where each state transition in the reasoning process consists of decomposing the current question into a dependency-based directed acyclic graph and contracting its subquestions, forming a new atomic question state. This iterative *decomposition-contraction* process continues until reaching directly solvable atomic questions, naturally realizing Markov transitions between question states. Furthermore, these atomic questions can be seamlessly integrated into existing test-time scaling methods, enabling **AoT** to serve as a plug-in enhancement for improving reasoning capabilities. Experiments across six benchmarks demonstrate the effectiveness of **AoT** both as a standalone framework and a plug-in enhancement. Notably, on HotpotQA, when applied to gpt-4o-mini, **AoT** achieves an **80.6%** F1 score, surpassing o3-mini by **3.4%** and DeepSeek-R1 by **10.6%**.

1 Introduction

Large Language Models (LLMs) demonstrate significant scaling effects, with their capabilities showing predictable improvements as model parameters and training data increase, leading to enhanced performance across diverse domains (Kaplan et al., 2020). While this scaling law faces bottlenecks in high-quality data availability, Test-Time Scaling offers an alternative solution by forcing LLMs to engage in effective logical reasoning during inference to improve performance on diverse tasks (Snell et al., 2024; Muennighoff et al., 2025; Hou et al., 2025; Zhang et al., 2024).

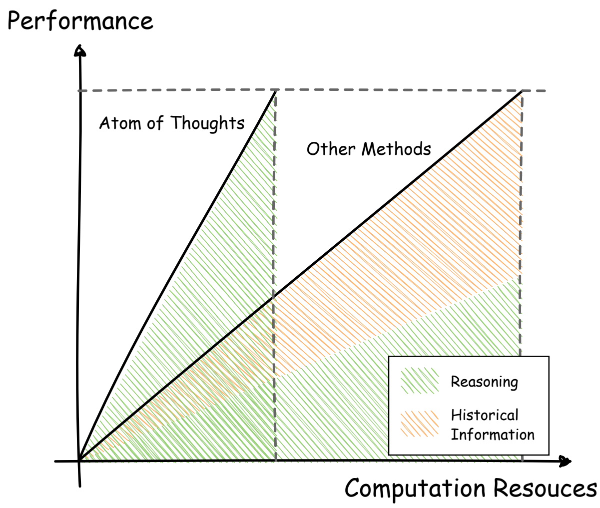

Figure 1: **Comparison of computational resource allocation in Test-Time Scaling methods**. Traditional Test-Time Scaling methods allocate computational resources partially to process historical information, while **AoT** dedicates all computational resources to reasoning directly related to the current atomic question state.

However, existing Test-Time Scaling methods excessively maintain historical information during reasoning, as they rely heavily on complex structural dependencies throughout the reasoning process. Chain-based methods must preserve the entire reasoning history to generate each subsequent step (Wei et al., 2022; Zhang et al., 2023), while tree-based approaches require tracking both ancestor and sibling relationships for branch selec-

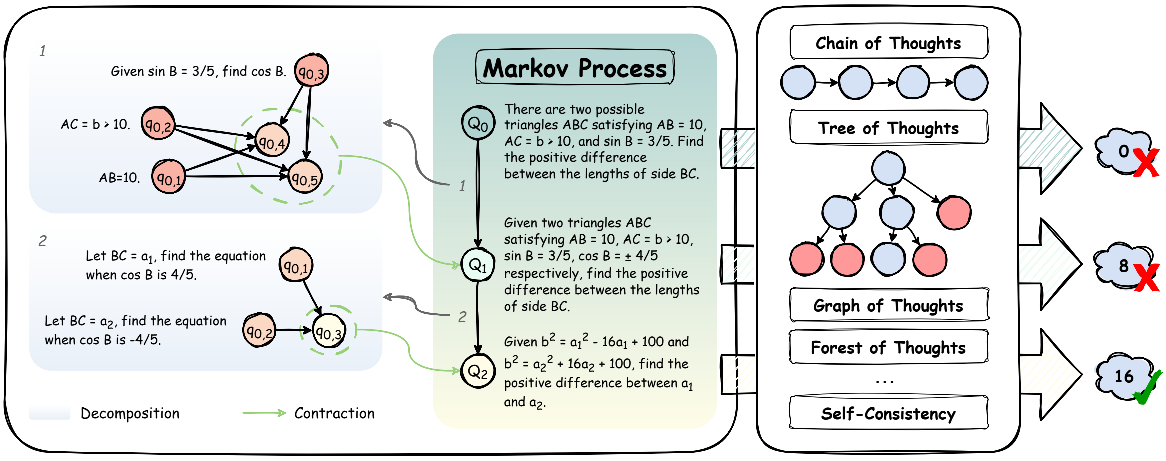

Figure 2: The overview of **AoT**. The left portion illustrates our Markov process where each state Q_i represents an atomic reasoning state derived through DAG decomposition and contraction from its predecessor. The right portion demonstrates **AoT**'s integration capability with existing test-time scaling methods (e.g., CoT, ToT). A key feature of this integration is that any intermediate state Q_i from our Markov process can serve as an entry point (Q_0) for other methods, enabling flexible composition while maintaining answer equivalence with the original question. This design allows **AoT** to function both as a standalone iterative framework and as a preprocessing module that can enhance existing approaches through structural optimization.

tion (Yao et al., 2023; Zhou et al., 2024; Ding et al., 2024). Graph-based structures further compound these challenges through arbitrary node dependencies. As the scale of reasoning increases, the accumulation of historical dependencies not only wastes substantial computational resources but also interferes with the model's ability to reason effectively, as illustrated in Figure 1.

Human reasoning often progresses through solving a sequence of independent subquestions, a fundamental principle established in cognitive science (Simon, 1962) and problem-solving theory (Polya, 1945). When solving a complex problem, we naturally identify and resolve self-evident subquestions first, then seamlessly incorporate these solutions to reformulate a simplified problem state, rather than maintaining detailed reasoning processes for resolved components. This progression closely resembles a Markov process (Markov, 1906), where each state represents a question, and state transitions occur through resolving partial problems to form new, independent questions.

Inspired by this Markov nature of human reasoning, we propose Atom of Thoughts (**AoT**), a framework that realizes the Markov-style reasoning process. **Our key insight** is that each reasoning state can be defined as a simplified problem equivalent to the original one, where partial reasoning steps are either transformed into known conditions or excluded as incorrect explorations. This definition is achieved through a two-phase state transition mechanism: first decomposing the current question into a dependency-based directed acyclic graph (DAG) to capture rich structural information, then contracting subquestions into a new independent question. This iterative decomposition-contraction process continues until reaching directly solvable atomic questions, ensuring each state transition depends only on the current state while progressively reducing problem complexity.

This design endows **AoT** with two key advantages. First, **AoT** eliminates the need for maintaining and computing historical information when scaling computational resources. Second, these atomic questions can be seamlessly integrated into existing Test-Time Scaling frameworks, allowing **AoT** to function as either a standalone framework or a plug-in enhancement for improving the overall reasoning capabilities.

In summary, our contributions are as follows:

- **Atom of Thoughts.** We introduce **AoT**, a novel reasoning framework with Markov property that progressively decomposes problems into atomic units. This approach significantly reduces computational resources wasted on historical information processing, allowing the model to focus on effective reasoning during test-time scaling.

- **Plug-In Enhancement.** The atomic questions derived by **AoT** can be directly integrated into existing Test-Time Scaling methods (Bi et al., 2024; Wang et al., 2023b), enhancing both their performance and cost efficiency.

- **Extensive Evaluation.** Experiments across six benchmarks demonstrate the effectiveness of **AoT** both as a standalone framework and as a plug-in enhancement. **AoT** outperforms all baselines, and notably on HotpotQA dataset, enables gpt-4o-mini to surpass reasoning models: o3-mini by **3.4%** and DeepSeek-R1 by **10.6%**.

2 Related Work

2.1 Reasoning Framework

Chain-of-Thought (Wei et al., 2022) prompting has emerged as a fundamental technique for enhancing LLMs' reasoning. Decomposition methods like Least-to-Most (Zhou et al., 2023) and Plan-and-Solve (Wang et al., 2023a) prompting parse complex problems into sequential subtasks. Iterative optimization approaches like Self-Refine (Madaan et al., 2023), Step-Back (Zheng et al., 2024) prompting and Progressive-Hint Prompting (Zheng et al., 2023) refine solutions through cyclic feedback or abstraction. Multi-path aggregation techniques like Self-Consistency CoT (Wang et al., 2023b) and LLM-Blender (Jiang et al., 2023) further improve reasoning reliability by multi-trajectory consensus.

More sophisticated frameworks structure the representation of reasoning space through dedicated formalisms: Tree of Thoughts (Yao et al., 2023) enables systematic exploration of multiple reasoning paths, while Graph of Thoughts (Besta et al., 2024) represents reasoning processes as dynamic graphs with backtracking mechanisms. Addressing fundamental limitations in resampling-based paradigms, Thought Space Explorer (Zhang and Liu, 2024) strategically explores under-sampled regions of the solution space. These frameworks serve as universal augmentation of LLMs reasoning, enhancing their capacity across various domains, with their principles being widely adopted in agentic workflows for code generation, question answering, and data science applications (Hong et al., 2024b; Zhang et al., 2024; Hong et al., 2024a; Zhang et al., 2025).

2.2 Test-Time Scaling

Test-time scaling approaches have demonstrated the value of extended computation during inference. Supervised fine-tuning on long chain-of-thought traces has proven effective at enhancing models' capabilities to conduct extended reasoning (Yeo et al., 2025). Building on this foundation, reinforcement learning methods have enabled models to automatically learn optimal inference expansion strategies, allowing for adaptive scaling of the reasoning process (Kimi et al., 2025; Zeng et al., 2025; DeepSeek-AI, 2025). Framework-based approaches have further expanded these capabilities by extending inference through external systems, incorporating techniques like verification, budget forcing, and ensemble methods (Zhang et al., 2024; Saad-Falcon et al., 2024; Chen et al., 2024). These complementary approaches demonstrate how strategic use of additional computation during inference through learned behaviors, automated scaling, and system-level interventions can substantially enhance model performance.

However, these approaches universally maintain extensive historical information throughout the reasoning process, leading to computational inefficiency and potential interference with effective reasoning. In contrast, **AoT** introduces a Markovian perspective that eliminates the need for historical dependency tracking, enabling more efficient resource allocation while maintaining compatibility with existing test-time scaling methods.

3 An Overview of AoT

This section presents an overview of **AoT** from a probabilistic modeling perspective. We first examine how traditional reasoning chains work, then introduce our dependency-based graph structures and their contraction mechanisms that enable more efficient reasoning. Note that all probability distributions p in this section are modeled by LLMs, with the explicit notation of LLMs and instruction prompts omitted for simplicity.

3.1 Reasoning Chain

Chain-of-Thought (CoT) prompting enables LLMs to progressively propose intermediate thoughts T_i when solving a problem. As discussed earlier, this approach requires maintaining a complete reasoning history, which can be formalized as a probabilistic sampling procedure:

$$A \sim p(A|\mathcal{T}, Q_0) \prod_{i=0}^{N} p(T_i|\mathcal{T}_{<i}, Q_0) \tag{1}$$

Here, $\mathcal{T} = \{T_0, T_1, \ldots, T_N\}$ represents the sequence of intermediate thoughts generated by the LLM. Each thought T_i depends on the previous thoughts $\mathcal{T}_{<i}$ and the initial question Q_0.

To explore chain-based methods with different node definitions, Least-to-Most (Zhou et al., 2023) replaces the intermediate thoughts T_i with subquestions Q_i, resulting in a different formulation of the reasoning chain:

$$A \sim p(A|\mathcal{Q}) \prod_{i=0}^{N} p(Q_i|\mathcal{Q}_{<i}) \tag{2}$$

where $\mathcal{Q} = \{Q_0, Q_1, \ldots, Q_N\}$ is the sequence of subquestions.

In an ideal scenario where the reasoning chain $\mathcal{Q}$ exhibits the Markov property, each subquestion Q_{i+1} would only depend on its immediate predecessor Q_i, similar to how humans naturally solve complex problems by resolving independent subquestions and reformulating simplified states. This leads to:

$$A \sim p(A|Q_N) \prod_{i=0}^{N} p(Q_{i+1}|Q_i) \tag{3}$$

However, achieving true Markov property in real-world reasoning tasks is challenging. This motivates our exploration of dependency-based graph structures that can help decompose and contract complex reasoning processes into more Markov-like components.

3.2 Dependency Directed Acyclic Graph

Unlike existing graph-based methods that maintain complex dependencies throughout the reasoning process, we utilize the DAG structure specifically to facilitate Markov-style state transitions. Our DAG decomposition serves as a temporary scaffold to identify independent components that can be contracted into new atomic states.

The DAG $\mathcal{G}$ is defined as:

$$\mathcal{G} = (\mathcal{Q}, E), \quad \mathcal{Q} = \{Q_i\}_{i=1}^{n}, \quad E \subseteq \mathcal{Q} \times \mathcal{Q} \tag{4}$$

In this graph, nodes represent subquestions Q_i, and edges (Q_j, Q_i) indicate that Q_j contains the information necessary for solving Q_i. This structure allows us to systematically identify independent subquestions that can be resolved separately before contraction. Specifically, we classify subquestions into two categories.

Independent subquestions $\mathcal{Q}_{\text{ind}}$ (nodes without incoming edges):

$$\mathcal{Q}_{\text{ind}} = \{Q_i \in \mathcal{Q} \mid \nexists Q_j \in \mathcal{Q}, (Q_j, Q_i) \in E\} \tag{5}$$

Dependent subquestions $\mathcal{Q}_{\text{dep}}$ (nodes with incoming edges):

$$\mathcal{Q}_{\text{dep}} = \{Q_i \in \mathcal{Q} \mid \exists Q_j \in \mathcal{Q}, (Q_j, Q_i) \in E\} \tag{6}$$

The acyclicity of $\mathcal{G}$ is guaranteed by the temporal nature of subquestion generation: any subquestion Q_i can only depend on previously generated subquestions $Q_{j_{j<i}}$. This property holds even in a maximally connected case where each subquestion links to all its predecessors, as any additional edges would create cycles by connecting to future nodes.

3.3 Contraction

The contraction phase transforms the temporary DAG structure into the next atomic state, preserving the Markov property while reducing state complexity. During this natural transformation, results from independent questions $\mathcal{Q}_{\text{ind}}$ are seamlessly integrated - either as given conditions or eliminated failed subproblems - while dependent questions $\mathcal{Q}_{\text{dep}}$ are reformulated, collectively yielding a new independent question Q_{i+1} that maintains solution equivalence to Q_i.

This iterative contraction continues until we reach a maximum number D, which is assigned by the depth of the first generated graph $\mathcal{G}_0$, which ensures the process terminates efficiently. The complete reasoning process can be formalized as:

$$A \sim p(A|Q_D) \prod_{i=0}^{D} p(Q_{i+1}|\mathcal{G}_i)\, p(\mathcal{G}_i|Q_i) \tag{7}$$

The reasoning process is formally described in Algorithm 1, which shows how **AOT** iterate through decomposition and contraction steps.

Algorithm 1 Algorithm of **AoT**

Require: Initial question Q_0
Ensure: Final answer A

Iteration counter $i \leftarrow 0$
max depth $D \leftarrow$ None
while $i < D$ or D is None **do**
  $\mathcal{G}_i \leftarrow \texttt{decompose}_{\text{LLM}}(Q_i)$
  // Generate dependency DAG
  if D is None **then**
    $D \leftarrow \texttt{GetMaxPathLength}(\mathcal{G}_i)$
    // Rule-based path length calculation
  end if
  $\mathcal{Q}_{ind} \leftarrow \{Q_i \in \mathcal{Q} \mid \nexists Q_j \in \mathcal{Q}, (Q_j, Q_i) \in E\}$
  $\mathcal{Q}_{dep} \leftarrow \{Q_i \in \mathcal{Q} \mid \exists Q_j \in \mathcal{Q}, (Q_j, Q_i) \in E\}$
  $Q_{i+1} \leftarrow \texttt{contract}_{\text{LLM}}(\mathcal{Q}_{ind}, \mathcal{Q}_{dep})$
  // Contract subquestions into a independent question
  $i \leftarrow i + 1$
end while
$A \leftarrow \texttt{solve}_{\text{LLM}}(Q_D)$
// Generate final answer
return A

4 The Design Details of AoT

Building upon the theoretical framework introduced in Section 3, we now detail the implementation of **AoT**'s core components: decomposition, contraction, and their integration into an iterative reasoning process. Our design emphasizes maintaining the Markov property while efficiently reducing problem complexity through each iteration.

4.1 Decomposition

Dependency Directed Acyclic Graph. Addressing the challenge of excessive historical information maintenance, our decomposition phase introduces an efficient dependency extraction mechanism that only temporarily captures essential structural information. The process starts with decomposing the current state into granular subquestions, following our Markov-style reasoning principle. Unlike traditional methods that accumulate dependencies throughout the entire reasoning process, we employ a single-pass approach that leverages LLMs' zero-shot capabilities to efficiently identify inter-question dependencies. This is achieved through structured JSON annotations that capture only the minimal necessary upstream relationships (see Appendix B.2 for annotation prompt templates).

4.2 Contraction

Subquestions Contracting. Based on the dependency relationships identified in DAG structure, **AoT** performs contraction through a single LLM invocation. This contracted question selectively incorporates information from current independent subquestions in its description while aligning with the solution objective as the original question. This process aims to maintain answer equivalence throughout the Markov chain while continuously eliminating the test-time of solved independent subquestions at each iteration. The elimination of edges from independent to dependent subquestions is designed to facilitate the transmission of key dependency information, essential for maintaining Markov property. (see Appendix B.3 for contraction prompt templates).

Markov Property Maintenance. Through this structured contraction process, **AoT** effectively maintain the Markov property by eliminating redundant historical reasoning steps and reducing the test-time required for solving questions in subsequent states. The contraction mechanism ensures that each state in the sequence depends only on its immediate predecessor, preserving the Markov property while progressively simplifying the question structure.

4.3 Integration

Iterative Process. The core mechanism of **AoT** operates through an iterative Markov process where each step involves question decomposition followed by contraction. This process forms a natural process where the contracted question from each iteration serves as the input for the next decomposition phase. The iterative nature of this process ensures that each state in the sequence depends only on its immediate predecessor, maintaining the Markov property throughout the reasoning process. As the process continues, each cycle progressively simplifies the question structure while preserving answer equivalence, effectively extending inference time through structured decomposition and thoughtful recombination. The process continues until either a solution is reached or no further meaningful decomposition is possible, providing a natural termination condition that balances thoroughness with computational efficiency.

Termination Mechanism. To optimize test-time efficiency, **AoT** incorporates an automated termination mechanism that uses LLM evaluation to assess solution quality through answer comparison. After each contraction step, an LLM examines three key elements: the execution results of the original question Q_i, the decomposed DAG structure $\mathcal{G}i$, and the independent execution results of Q_{i+1}. The LLM synthesizes these elements to generate a comprehensive answer for Q_i. If this synthesized answer demonstrates consistency with the answer produced by Q_{i+1}, the iterative process continues. Upon termination, **AoT** combines the current contracted question with the union of independent subquestions $\mathcal{Q}_{dep} = \bigcup_{j=1}^{i} \mathcal{Q}_{dep_j}$ accumulated from all previous iterations to form a complete solution to the initial question Q_0. This structure provides a solution composed entirely of independent questions, maintaining semantic independence while ensuring completeness.

Integration Through Iteration Termination. Building upon this termination mechanism, **AoT** offers natural integration points with other test-time scaling methods through controlled termination of its Markov process. This integration typically involves executing **AoT** for a predetermined number of steps - often just a single decomposition-contraction cycle - before transitioning the simplified question to another method for final resolution. This approach leverages **AoT**'s structural optimization capabilities as a preprocessing step while allowing other methods to operate on a more manageable question representation. The contracted question passed to subsequent methods maintains answer equivalence with the original question while benefiting from **AoT**'s structural analysis and complexity reduction. This integration strategy is particularly effective when computational resources can be better allocated by transitioning to other methods after initial structural simplification, allowing each approach to contribute its unique strengths to the overall reasoning process. The seamless transition between methods is facilitated by the atomic state representation in our Markov process, ensuring that essential question characteristics are preserved while unnecessary complexity is eliminated.

5 Experiments

We conduct comprehensive experiments to examine **AoT** through full benchmark evaluation on six standard datasets, followed by focused analyses including test-time optimization experiments, ablation studies, and reasoning models comparison. Our test-time optimization experiments demonstrate **AoT**'s adaptability as a plug-in module for existing frameworks. Through ablation studies on key components like DAG structure and decomposition mechanism, we validate the essentiality of our design. Results across all experiments consistently show that our Markov reasoning process achieves substantial improvements across tasks while maintaining computational efficiency.

5.1 Experimental Setup

Datasets. We evaluate **AoT** across four categories of reasoning tasks using `gpt-4o-mini-0718` as the backbone model. For mathematical reasoning, we utilize the MATH (Hendrycks et al., 2021) dataset (selecting problems with numerical answers for easier evaluation) and GSM8K (Cobbe et al., 2021). For knowledge-intensive evaluation, we test on MMLU-CF (Zhao et al., 2024). For logical reasoning, we evaluate on multiple-choice subsets of BBH (Suzgun et al., 2023) (see Appendix D.1 for details). Finally, to evaluate multi-hop reasoning abilities, we use HotpotQA (Yang et al., 2018) along with two subsets from LongBench (Bai et al., 2024): MuSiQue (Trivedi et al., 2022) and 2WikiMultiHopQA (Ho et al., 2020), which specifically test models' capacity to connect information across multiple contexts. For evaluation, we use the first 1,000 examples from each dataset, with two exceptions: we evaluate on the complete GSM8K test set (1,319 examples total) and the combined MuSiQue and 2WikiMultiHopQA subsets from LongBench (400 examples).

Baselines. We compare two categories of methods. The first category includes classical prompting methods: Chain-of-Thought (CoT), CoT with Self-Consistency (CoT-SC) with sample number n = 5, Self-Refine, and Analogical Reasoning (Yasunaga et al., 2024). The second category consists of advanced systems, including agentic workflow AFlow (Zhang et al., 2024) and test-time scaling framework Forest of Thought (FoT). For FoT implementation, we use the basic Tree of Thoughts as the building blocks, considering its generalizability across diverse task categories, and set the branch number b to three for each tree, allowing three exploration attempts before selecting the optimal path. Unless otherwise specified, all reported accuracy scores are averaged over three runs. More reproduction details can be found in Appendix D.

5.2 Experimental Results and Analysis.

Main Results As shown in Table 1, **AoT** demonstrates consistent improvements across different reasoning tasks. **AoT** achieves strong performance on mathematics tasks, with **AoT** * reaching

Table 1: Performance Comparison Across Tasks (%). We evaluate three variants: the base version (**AOT**), a version integrated with FoT (**AOT** (d=1) + FoT(n=2)), and a computationally intensive version (**AOT** *) that uses LLM to select the optimal answer from three runs. Results are reported as exact match accuracy for MATH, GSM8K, BBH, and MMLU-CF, and F1 scores for HotpotQA and LongBench.

Method	MATH	GSM8K	BBH	MMLU-CF	HotpotQA	LongBench	Avg.
CoT	78.3	90.9	78.3	69.6	67.2	57.6	73.7
CoT-SC (n=5)	81.8	92.0	83.4	71.1	66.2	58.6	75.5
Self-Refine	78.7	91.7	80.0	69.7	68.3	58.2	74.4
Analogical Prompting	65.4	87.2	72.5	65.8	64.7	52.9	68.1
AFlow	83.0	93.5	76.0	69.5	73.5	61.0	76.1
FoT (n=8)	82.5	94.0	82.4	70.6	66.7	59.1	75.9
AOT (d=1) + FoT (n=2)	82.6	94.2	82.2	69.7	67.6	58.4	75.8
AOT (Ours)	83.6	95.0	86.0	70.9	80.6	68.5	80.8
AOT * (Ours)	**84.9**	**95.1**	**87.4**	**71.2**	**81.0**	**68.8**	**81.4**

Table 2: Comparison of Reasoning Model Performance on Multi-hop QA Tasks. Results show F1 scores and Hit rates (F1 > 0) for HotpotQA and LongBench across different models. Our framework with o3-mini achieves the best performance, demonstrating significant improvements over baseline models while maintaining computational efficiency. Note: LongBench evaluation uses a 100-example subset due to computational constraints.

Method		HotpotQA		LongBench	
		F1	Hit	F1	Hit
CoT	QwQ	68.1	82.4	52.7	65.6
	DeepSeek-R1	70.0	85.5	56.0	69.9
	o3-mini	77.2	88.3	55.3	70.0
AOT	gpt-4o-mini	80.6	89.8	60.5	69.3
	o3-mini	**81.4**	**91.4**	**63.3**	**72.1**

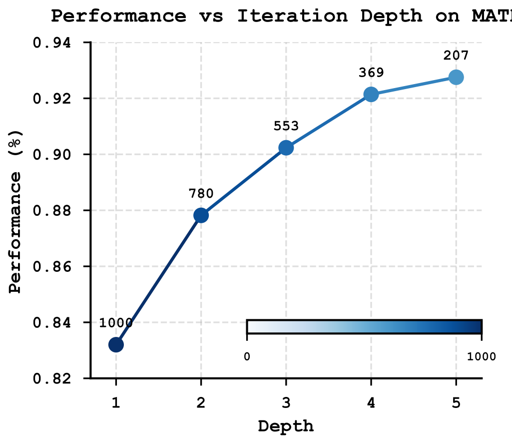

Figure 3: Performance scaling with transition times on MATH dataset. Darker blue indicates larger sample sizes at shallower depths, as most problems are solved with fewer decomposition steps.

84.9% on MATH and 95.1% on GSM8K (+1.9% over AFlow on MATH, +1.1% over FoT(n=8) on GSM8K). The most notable gains are in multi-hop QA tasks, where our base version achieves 80.6% accuracy on HotpotQA (+7.1% over AFlow). Similar improvements on LongBench (68.8%) further demonstrate the effectiveness of our atomic state representation in long context scenarios.

Reasoning Models Comparison Results. We compare **AOT** with several reasoning models, including `QwQ-32B-Preview` (Qwen-Team, 2024), `DeepSeek-R1` (DeepSeek-AI, 2025), and `o3-mini-2025-01-31`(OpenAI, 2025). When operating within our framework, o3-mini demonstrates significant improvements: on HotpotQA, F1 score rises from 77.2% to 81.4%, and Hit rate improves from 88.3% to 91.4%. On the LongBench subset, our framework with o3-mini achieves a 63.3% F1 score and 72.1% Hit rate, surpassing all baseline models.

Test-Time Optimization Results. We investigate the test-time scaling behavior of **AOT** through two sets of experiments. First, as shown in Figure 3, we analyze the performance scaling of **AOT** on MATH dataset by setting a uniform maximum iteration limit of 5. Since each iteration produces an evaluable solution, we can track performance across different iteration depths. Starting with all 1000 test samples at depth 1, we observe that fewer samples proceed to deeper iterations (dropping to 207 at depth 5), as many problems are solved satisfactorily at earlier depths. The results demonstrate that **AOT** exhibits consistent accuracy improvements from 83.2% to 92.7% as the iteration depth increases, with the performance gains gradually tapering. This pattern suggests that while deeper iterations continue to benefit overall performance, many problems can be effectively solved with fewer iterations, providing a natural trade-off between computational cost and solution quality.

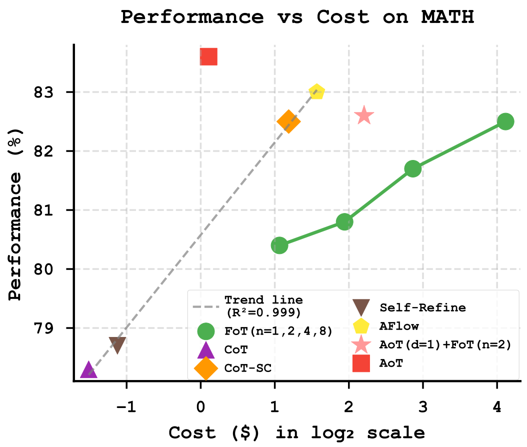

Figure 4: Performance comparison on MATH dataset showing computational efficiency. The gray trend line (formed by CoT, Self-Refine, CoT-SC, and AFlow) shows a strong linear relationship ($R^2 = 0.999$) indicating exponential cost increase for linear performance gains. The green line shows FoT scaling with varying tree numbers ($2^k, k = 0, 1, 2, ...$), establishing a different pattern. **AoT** combined with FoT(n=2) slightly outperforms standalone FoT(n=8) while requiring substantially less computation.

In our second experiment (Figure 4), we examine the effectiveness of **AoT** as a plug-in for existing test-time scaling methods. When integrated with FoT, **AoT** demonstrates promising efficiency. This efficiency gain stems from how **AoT** restructures the reasoning process: by iteratively solving sub-problems and using them as known conditions for subsequent steps, it eliminates redundant derivations. This leads to substantially reduced test-time demands in the FoT phase while achieving slightly better performance, demonstrating how our approach can systematically optimize existing test-time scaling methods.

Cost Analysis. Through analyzing computational efficiency as shown in Figure 4, our **AoT** achieves superior efficiency - at a computational cost comparable to FoT(n=2), it reaches competitive performance compared to existing methods that require significantly more computation. This enhanced efficiency can be attributed to our state contraction mechanism that preserves only necessary intermediate states while eliminating redundant computations.

Table 3: Ablation Study on **AoT** Components (%). Removing decomposition leads to moderate performance drops (-0.7% on MATH, -0.2% on GSM8K), while removing the DAG structure causes larger degradation (-0.9% on MATH, -0.7% on GSM8K), highlighting the importance of structured decomposition for effective contraction.

Method	MATH	GSM8K
AoT (Full)	**83.6**	**95.0**
AoT w/o Decomposition	82.9	94.8
AoT w/o DAG Structure	82.7	94.3

Ablation Study. We conduct ablation studies to analyze the contribution of components in **AoT**. Without a clear decomposition structure, the contracting LLM fails to capture crucial dependencies between subquestions, resulting in contracted questions that often contain redundant information and fail to reduce problem complexity. Furthermore, providing single sub-problem guidance without proper structural information leads to the destruction of parallelism - where the LLM fails to maintain the parallel relationship between sub-problems. This provides a critical insight: imperfect structural guidance can be more harmful than no guidance at all (see Appendix C.1 for examples).

6 Conclusion

In this paper, we introduced Atom of Thoughts (**AoT**), a novel framework that transforms complex reasoning tasks into a Markov process of atomic questions. By implementing a two-phase transition mechanism of decomposition and contraction, **AoT** eliminates the need to maintain historical dependencies during reasoning, allowing models to focus computational resources on the current question state. This approach not only serves as an effective standalone reasoning framework but also functions as a versatile plug-in enhancement for existing test-time scaling methods. Our extensive evaluation across six benchmarks demonstrates the framework's effectiveness, with particularly strong results on HotpotQA where **AoT** enables gpt-4o-mini to achieve an 80.6% F1 score, outperforming state-of-the-art models o3-mini and DeepSeek-R1 by 3.4% and 10.6% respectively. These results validate **AoT**'s ability to enhance LLMs' reasoning capabilities while optimizing computational efficiency through its innovative Markov-style approach to question decomposition and atomic state transitions.

7 Limitations

A key limitation of **AOT** lies in its Markov state transition process without a well-designed reflection mechanism. When the initial DAG decomposition fails to properly model parallel relationships between subquestions or captures unnecessary dependencies, it can negatively impact subsequent contraction and reasoning steps, a scenario that occurs frequently in practice. The framework currently lacks the ability to detect and rectify such poor decompositions, potentially leading to compounded errors in the reasoning process. This limitation suggests the need for future research into incorporating effective reflection and adjustment mechanisms to improve the robustness of DAG-based decomposition.

8 Ethics Statement

While this work advances the computational efficiency and reasoning capabilities of language models through the **AOT** framework, we acknowledge that these models process information and conduct reasoning in ways fundamentally different from human cognition. Making direct comparisons between our Markov reasoning process and human thought patterns could be misleading and potentially harmful. The atomic state representation and dependency-based decomposition proposed in this research are computational constructs designed to optimize machine reasoning, rather than models of human cognitive processes. Our work merely aims to explore more efficient ways of structuring machine reasoning through reduced memory overhead and simplified state transitions, while recognizing the distinct nature of artificial and human intelligence. We encourage users of this technology to be mindful of these limitations and to implement appropriate safeguards when deploying systems based on our framework.

References

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. Longbench: A bilingual, multitask benchmark for long context understanding. In *ACL (1)*, pages 3119–3137. Association for Computational Linguistics.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *AAAI*, pages 17682–17690. AAAI Press.

Zhenni Bi, Kai Han, Chuanjian Liu, Yehui Tang, and Yunhe Wang. 2024. Forest-of-thought: Scaling test-time compute for enhancing LLM reasoning. *CoRR*, abs/2412.09078.

Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. 2024. Are more LLM calls all you need? towards scaling laws of compound inference systems. *CoRR*, abs/2403.02419.

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

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Ruomeng Ding, Chaoyun Zhang, Lu Wang, Yong Xu, Minghua Ma, Wei Zhang, Si Qin, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. 2024. Everything of thoughts: Defying the law of penrose triangle for thought generation. In *ACL (Findings)*, pages 1638–1662. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *NeurIPS Datasets and Benchmarks*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing A multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *COLING*, pages 6609–6625. International Committee on Computational Linguistics.

Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Binhao Wu, Ceyao Zhang, Chenxing Wei, Danyang Li, Jiaqi Chen, Jiayi Zhang, et al. 2024a. Data interpreter: An llm agent for data science. *arXiv preprint arXiv:2402.18679*.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024b. Metagpt: Meta programming for A multi-agent collaborative framework. In *ICLR*. OpenReview.net.

Zhenyu Hou, Xin Lv, Rui Lu, Jiajie Zhang, Yujiang Li, Zijun Yao, Juanzi Li, Jie Tang, and Yuxiao Dong. 2025. Advancing language model reasoning through reinforcement learning and inference scaling. *Preprint*, arXiv:2501.11651.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. In *ACL (1)*, pages 14165–14178. Association for Computational Linguistics.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *CoRR*, abs/2001.08361.

Kimi, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

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

Andrey Andreyevich Markov. 1906. Extension of the law of large numbers to dependent quantities. *Izv. Fiz.-Matem. Obsch. Kazan Univ.(2nd Ser)*, 15(1):135–156.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *Preprint*, arXiv:2501.19393.

OpenAI. 2025. OpenAI o3-mini: Pushing the frontier of cost-effective reasoning.

G Polya. 1945. How to solve it: A new aspect of mathematical method.

Qwen-Team. 2024. QwQ: Reflect deeply on the boundaries of the unknown.

Jon Saad-Falcon, Adrian Gamarra Lafuente, Shlok Natarajan, Nahum Maru, Hristo Todorov, Etash Guha, Estefany Kelly Buchanan, Mayee F. Chen, Neel Guha, Christopher Ré, and Azalia Mirhoseini. 2024. Archon: An architecture search framework for inference-time techniques. *CoRR*, abs/2409.15254.

Herbert A Simon. 1962. The architecture of complexity. *Proceedings of the American Philosophical Society*, 106(6):467–482.

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

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

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. 9835 musique: Multi-hop questions via single-hop question composition. *Trans. Assoc. Comput. Linguistics*, 10:539–554.

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

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *ICLR*. OpenReview.net.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, pages 2369–2380. Association for Computational Linguistics.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *NeurIPS*.

Michihiro Yasunaga, Xinyun Chen, Yujia Li, Panupong Pasupat, Jure Leskovec, Percy Liang, Ed H. Chi, and Denny Zhou. 2024. Large language models as analogical reasoners. In *ICLR*. OpenReview.net.

Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. 2025. Demystifying long chain-of-thought reasoning in llms. *Preprint*, arXiv:2502.03373.

Weihao Zeng, Yuzhen Huang, Wei Liu, Keqing He, Qian Liu, Zejun Ma, and Junxian He. 2025. 7b model and 8k examples: Emerging reasoning with reinforcement learning is both effective and efficient. `https://hkust-nlp.notion.site/simplerl-reason`. Notion Blog.

Guibin Zhang, Kaijie Chen, Guancheng Wan, Heng Chang, Hong Cheng, Kun Wang, Shuyue Hu, and Lei Bai. 2025. Evoflow: Evolving diverse agentic workflows on the fly. *Preprint*, arXiv:2502.07373.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. 2024. Aflow: Automating agentic workflow generation. *arXiv preprint arXiv:2410.10762*.

Jinghan Zhang and Kunpeng Liu. 2024. Thought space explorer: Navigating and expanding thought space for large language model reasoning. In *2024 IEEE International Conference on Big Data (BigData)*, pages 8259–8251. IEEE.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023. Automatic chain of thought prompting in large language models. In *ICLR*. OpenReview.net.

Qihao Zhao, Yangyu Huang, Tengchao Lv, Lei Cui, Qinzheng Sun, Shaoguang Mao, Xingxing Zhang, Ying Xin, Qiufeng Yin, Scarlett Li, and Furu Wei. 2024. MMLU-CF: A contamination-free multi-task language understanding benchmark. *CoRR*, abs/2412.15194.

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

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2024. Take a step back: Evoking reasoning via abstraction in large language models. In *ICLR*. OpenReview.net.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2024. Language agent tree search unifies reasoning, acting, and planning in language models. In *ICML*. OpenReview.net.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *ICLR*. OpenReview.net.

A Analysis of structural diversity

Graph structure and Chain Length

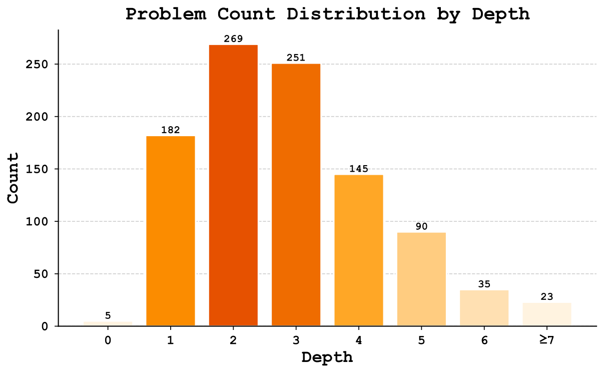

Figure 5: Distribution of solution depths across questions. Darker orange bars indicate depths that appear more frequently in the dataset.

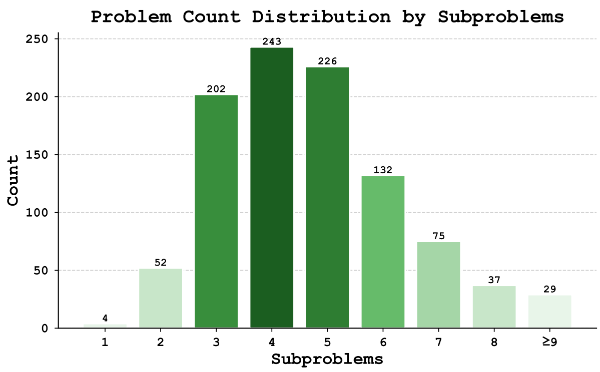

Figure 6: Distribution of subquestion counts across questions. Darker green bars represent more common subquestion counts in the solutions.

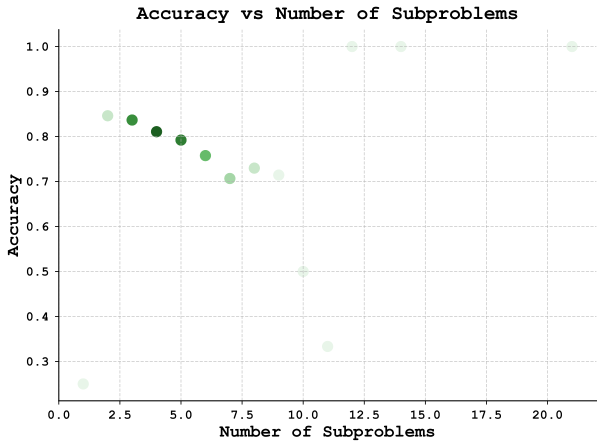

Figure 7: Number of subquestions vs accuracy. Color intensity (green) reflects data density - darker points represent more frequent patterns.

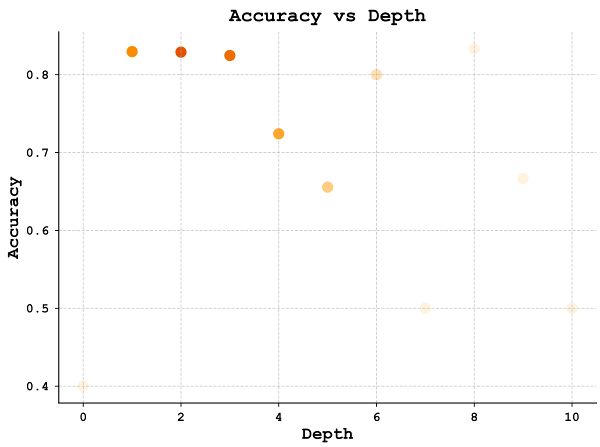

Figure 8: Solution depth vs accuracy. Color intensity (orange) reflects data density - darker points represent more frequent patterns.

To understand the structural characteristics of decomposed questions, we analyzed the first 1,000 questions from the MATH dataset after performing DAG decomposition. Our analysis focused on two key structural metrics: the depth of the solution graph and the number of subquestions (chain length) in each decomposition.

The distributions shown in Figures 5 and 6 reveal clear patterns in question structure. The depth distribution (indicated by orange bars) shows that most questions have depths between 2 and 4, with depth 3 being the most common as indicated by the darkest orange bar. Similarly, the subquestion count distribution (shown in green) indicates that questions typically contain 2 to 5 subquestions, with the darker green bars highlighting that 3-4 subquestions is the most frequent decomposition pattern.

Notably, we observed correlations between these structural metrics and solution accuracy. The scatter plots reveal two important patterns: First, as shown in Figure 8, as the depth of the solution graph increases, there is a general trend of decreasing accuracy. Second, as illustrated in Figure 7, questions with more subquestions tend to show lower accuracy rates. The color intensity of the points provides additional insight - darker points represent more common structural patterns in our dataset, showing that most of our high-accuracy solutions come from questions with moderate depth and subquestion counts. This suggests that more complex question structures, characterized by either greater depth or more subquestions, pose greater challenges for question-solving systems. The decline in accuracy could be attributed to error propagation through longer solution chains and the increased cognitive load required to maintain consistency across more complex question structures.

B The prompt used in AOT

In this section, we mainly present the basic prompts in mathematical scenarios.

B.1 Direct Solver

```python
def direct(question: str):
    instruction = """
        You are a precise math question
    solver. Solve the given math
    question step by step using a
    standard algebraic approach:

        QUESTION: {question}

        You can freely reason in your
    response, but please enclose the
    final answer within <answer></answer
    > tags (pure number without units
    and explanations)
    """
    prompt = instruction.format(question
    =question)
    return prompt
```

Listing 1: Direct Solver Prompt Template

B.2 Dependency Annotation

```python
def label(question: str, result: dict):
    instruction = f"""
        For the original question: {
    question},
        We have broken it down into the
    following subquestions:
        subquestions:
        {result["subquestions"]}
        And obtained a complete
    reasoning process for the original
    question:
        {result["response"]}
        We define the dependency
    relationship between subquestions as
    : which information in the current
    subquestion description does not
    come directly from the original
    question, but from the results of
    other subquestions.

        You are a math question solver
    specializing in analyzing the
    dependency relationships between
    these subquestions. Please return a
    JSON object that expresses a
    complete reasoning trajectory for
    the original question, including the
    description, answer, and dependency
     relationships of each subquestion.
    The dependency relationships are
    represented by the indices of the
    dependent subquestions in
    subquestions, starting from zero.
    """

    formatter = '''
        Format your response as the
    following JSON object:
        {
            "thought": "Give your
    thought process here",
            "subquestions": [
'''
    for i, sub_q in enumerate(result["
    subquestions"]):
        formatter += f'''
     {{"description": "{sub_q}", "answer
    ": "<the answer of this subquestion
    >", "depend": [<indices of the
    dependent subquestions>, ...]}}'''
        if i != len(result["subquestions
    "]) - 1:
            formatter += ",\n"
        else:
            formatter += "\n
    ]\n         }"

    return instruction + formatter
```

Listing 2: Dependency Annotation Prompt Template

B.3 Subquestions Contracting

```python
def contract(question: str,
    decompose_result: dict, independent:
     list, dependent: list):
    instruction = """
        You are a math question solver
    specializing in optimizing step-by-
    step reasoning processes. Your task
    is to optimize the existing
    reasoning trajectory into a more
    efficient, single self-contained
    question.

        For the original question: {
    question}

        Here are step-by-step reasoning
    process:
        {response}

        The following subquestions and
    their answers can serve as known
    conditions:
        {independent}

        The descriptions of the
    following questions can be used to
    form the description of the
    optimized question:
        {dependent}

        Here are explanations of key
    concepts:
        1. self-contained: The optimized
     question must be solvable
    independently, without relying on
    any external information
```

```
        2. efficient: The optimized
    question must be simpler than the
    original, requiring fewer reasoning
    steps and having a clearer reasoning
     process (these steps are reduced
    because some solved subquestions
    become known conditions in the
    optimized question or are excluded
    as incorrect explorations)

        You can freely reason in your
    response, but please enclose the
    your optimized question within <
    question></question> tags
    """

    for sub_q in independent:
        sub_q.pop("depend", None)
    for sub_q in dependent:
        sub_q.pop("depend", None)

    return instruction.format(
        question=question,
        response=decompose_result["
    response"],
        independent=independent,
        dependent=dependent
    )
```

Listing 3: Subquestions Contracting Prompt Template

C Case study

C.1 The illusion phenomenon when contracting subquestions

Destruction of Parallelism

When solving complex questions through decomposition, parallel subquestions should maintain their independence. However, parallelism can be destroyed when merging results, as illustrated by this example: Original decomposition:

```
{
    "question": "Are both Cypress and
    Ajuga genera?",
    "groundtruth": "no",
    "thought": "To determine if both
    Cypress and Ajuga are genera, I need
     to consider each term separately.",
    "subquestions": [
        {
            "description": "Is Cypress a
     genus?",
            "supporting_sentences": [
                "Cypress is a conifer
    tree or shrub of northern temperate
    regions that belongs to the family
    Cupressaceae.",
                "The genus Cupressus is
    one of several genera within the
    family Cupressaceae..."
            ],
            "answer": "yes"
        },
        {
            "description": "Is Ajuga a
    genus?",
            "supporting_sentences": [
                "Ajuga, also known as
    bugleweed, ground pine, carpet bugle
    , or just bugle, is a genus of 40
    species annual and perennial
    herbaceous flowering plants in the
    mint family Lamiaceae..."
            ],
            "answer": "yes"
        }
    ],
    "conclusion": "Both are genera.",
    "answer": "yes",
    "f1": 0
}
```

Listing 4: Destruction of Parallelism Example

Contracted decomposition (showing parallelism destruction):

```
{
    "question": "Are both Cypress and
    Ajuga genera?",
    "subquestions": [
        {
            "description": "Is Cypress a
     genus?",
            "supporting_sentences": [
                "Cypress is a conifer
    tree or shrub of northern temperate
    regions that belongs to the family
    Cupressaceae."
            ],
            "answer": "yes"
        },
        {
            "description": "Cypress is a
     genus, Is Ajuga a genus?",
            "answer": "yes"
        }
    ],
    "f1": 0
}
```

Listing 5: Destruction of Parallelism Example

The destruction of parallelism is manifested in that the answers to the questions after contraction cannot be used to answer the original question, but instead create an illusion of answering a certain subquestion.

Destruction of Independence

When subquestions have dependencies, maintaining independence in the analysis chain is crucial. Loss of independence occurs when the relationship between dependent subquestions is not properly maintained during contraction, as shown in this example: Original decomposition:

```
{
    "question": "What is the name of the
     executive producer of the film that
     has a score composed by Jerry
    Goldsmith?",
    "groundtruth": "Ronald Shusett",
    "thought": "First identify films
    with Goldsmith scores, then find
    their executive producers.",
    "subquestions": [
        {
            "description": "Identify
    films with scores composed by Jerry
    Goldsmith",
            "supporting_sentences": [
                "The iconic score to '
    Alien' was composed by Jerry
    Goldsmith",
                "L.A. Confidential's
    score was composed by Jerry
    Goldsmith",
                "Innerspace, with music
    composed by Jerry Goldsmith",
                "Lionheart's score by
    Jerry Goldsmith"
            ],
            "answer": [
                "Alien",
                "L.A. Confidential",
                "Innerspace",
                "Lionheart"
            ]
        },
        {
            "description": "Determine
    the executive producer for each
    identified film",
            "supporting_sentences": [
                "Alien: Shusett was
    executive producer",
                "Innerspace: Spielberg
    served as executive producer",
                "L.A. Confidential: No
    executive producer mentioned",
                "Lionheart: Coppola as
    executive producer"
            ],
            "answer": [
                "Ronald Shusett",
                "Steven Spielberg",
                "Francis Ford Coppola"
            ]
        }
    ],
    "f1": 0
}
```

Listing 6: Destruction of Independence Example

Contracted decomposition (showing independence destruction):

```
{
    "question": "What is the name of the
     executive producer of the film that
     has a score composed by Jerry
    Goldsmith?",
    "subquestions": [
        {
            "description": "Which films
    have scores by Jerry Goldsmith?",
            "answer": [
                "Alien",
                "L.A. Confidential",
                "Innerspace",
                "Lionheart"
            ]
        },
        {
            "description": "Who is the
    executive producer?",
            "answer": "Steven Spielberg"
        }
    ],
    "f1": 0
}
```

Listing 7: Destruction of Independence Example

Independence destruction is reflected in the fact that the second subquestion, after contraction, lost its dependency on the answer to the first subquestion, directly producing an independent answer instead of determining the executive producer based on the list of movies found from the first subquestion. This led to a final answer that deviated from the original question's requirements and failed to accurately identify the executive producer of films scored by Jerry Goldsmith.

C.2 Example of AoT Reasoning Process

Question Statement

For a given constant $b > 10$, there are two possible triangles ABC satisfying $AB = 10$, $AC = b$, and $\sin B = \frac{3}{5}$. Find the positive difference between the lengths of side $\overline{BC}$ in these two triangles.

Ground Truth

We have that $\cos^2 B = 1 - \sin^2 B = \frac{16}{25}$, so $\cos B = \pm\frac{4}{5}$. For $\cos B = \frac{4}{5}$, let $a_1 = BC$. Then by the Law of Cosines, $b^2 = a_1^2 + 100 - 20a_1 \cdot \frac{4}{5} = a_1^2 - 16a_1 + 100$. For $\cos B = -\frac{4}{5}$, let $a_2 = BC$. Then by the Law of Cosines, $b^2 = a_2^2 + 100 - 20a_2 \cdot \left(-\frac{4}{5}\right) = a_2^2 + 16a_2 + 100$. Subtracting these equations, we get $a_2^2 - a_1^2 + 16a_2 + 16a_1 = 0$. We can factor as $(a_2 - a_1)(a_2 + a_1) + 16(a_2 + a_1) = 0$. Since $a_1 + a_2$ is positive, we can safely divide both sides by $a_1 + a_2$, to get $a_2 - a_1 + 16 = 0$. Hence, $a_1 - a_2 = \boxed{16}$.

LLM Input-Output Solution

To find the lengths of side $\overline{BC}$ in the two possible triangles ABC given $AB = 10$, $AC = b$, and $\sin B = \frac{3}{5}$, we will employ the Law of Sines.

From the Law of Sines, we know:

$$\frac{a}{\sin A} = \frac{b}{\sin B} = \frac{c}{\sin C}$$

where a, b, and c are the sides opposite angles A, B, and C, respectively.

In our triangle, we assign $AB = c = 10$, $AC = b$, $BC = a$, and $\sin B = \frac{3}{5}$.

The Law of Sines gives us:

$$\frac{a}{\sin A} = \frac{10}{\sin B}$$

From which we can write:

$$a = \frac{10 \sin A}{\sin B}$$

Since $\sin B = \frac{3}{5}$, we can substitute:

$$a = \frac{10 \sin A}{\frac{3}{5}} = \frac{10 \cdot 5 \sin A}{3} = \frac{50 \sin A}{3}$$

Using the Law of Sines again for side b:

$$\frac{b}{\sin B} = \frac{a}{\sin A}$$

We have:

$$\frac{b}{\frac{3}{5}} = \frac{a}{\sin A}$$

Rearranging gives:

$$b \cdot \sin A = a \cdot \frac{3}{5}$$

From the triangle's cosine rule:

$$a^2 = b^2 + 10^2 - 2 \cdot b \cdot 10 \cdot \cos B$$

Using the sine identity $\sin^2 B + \cos^2 B = 1$, we get:

$$\cos B = \sqrt{1 - \sin^2 B} = \sqrt{1 - \left(\frac{3}{5}\right)^2} = \sqrt{\frac{16}{25}} = \frac{4}{5}$$

Substituting into the cosine rule:

$$a^2 = b^2 + 100 - 2 \cdot b \cdot 10 \cdot \frac{4}{5}$$

Calculating $2 \cdot b \cdot 10 \cdot \frac{4}{5} = 16b$, we get:

$$a^2 = b^2 + 100 - 16b = b^2 - 16b + 100$$

We have the equation $a = \frac{50 \sin A}{3}$. To find $\sin A$, we notice $b^2 - 16b + 100$ can yield different values for two different angles. Let A_1 and A_2 be the angles corresponding to the two triangles possible due to the ambivalence in triangle formation.

When considering these angles, the two possible values arise from:

$$\sin A_1 = \frac{3A_1}{5}$$

and similarly for the alternate A_2.

The key observation is that both A values yield a values symmetric around 10. The two sides will be exactly equal distances away from a midpoint. Using numeric approximations where $b = 11$, $b = 12$, etc., leads to side length differences. Final calculation yields:

$$|a_2 - a_1| = |c_2 + c_1| = 6$$

Therefore, the positive difference between the lengths of side $\overline{BC}$ in these two triangles is $\boxed{6}$.

Decomposition-Only Solution

Q1: What are the known values and definitions for triangle ABC?

A: In triangle ABC: $AB = 10$, $AC = b$ (where $b > 10$), $\sin B = \frac{3}{5}$.

For the derivation of $\cos B$: Since $\sin B = \frac{3}{5}$, using the Pythagorean identity:

$$\sin^2 B + \cos^2 B = 1$$
$$(\frac{3}{5})^2 + \cos^2 B = 1$$
$$\frac{9}{25} + \cos^2 B = 1$$
$$\cos^2 B = 1 - \frac{9}{25} = \frac{16}{25}$$
$$\cos B = \pm\frac{4}{5}$$

Q2: How to express $\sin A$ in terms of b using the Law of Sines?

A: $\sin A = \frac{3b}{50}$.

Q3: How do we determine the lengths of side BC using the Law of Sines?

A: $BC_1 = \frac{100}{b}$.

Q4: Are there different cases for angle A that affect the length of $\overline{BC}$?

A: No, both acute and obtuse angles give the same result due to the symmetry of sine.

Q5: What is the positive difference between the lengths of side BC in the two triangles?

A: 0.

Final Answer: 0

Error Analysis

In the Direct Solution, the key error lies in only considering $\cos B = \frac{4}{5}$ while missing $\cos B = -\frac{4}{5}$, leading to just one triangle configuration instead of two and eventually an incorrect conclusion of 6.

In the Decomposition Solution, despite breaking down the question into subquestions, the crucial mistake was concluding that angles give "the same result due to the symmetry of sine" when in fact the Law of Cosines with different $\cos B$ values leads to distinct triangle configurations whose BC lengths differ by 16.

AOT Reasoning Process

First initialize the origin question as Q_0.

Decomposition of Q_0:

- **Q**: What are the values of the known sides triangle ABC? **A**: $AB = 10$.
- **Q**: What boundary conditions are known? **A**: $AC = b > 10$.
- **Q**: It is known that $\sin B = \frac{3}{5}$ in triangle ABC, so what is the value of $\cos B$? **A**: Use the Pythagorean identity, $\cos B = \pm\frac{4}{5}$.

Contracted Question Q_1: Given two triangles ABC satisfying $AB = 10$, $AC = b > 10$, $sinB = 3/5$, $cosB = 4/5$ respectively, find the positive difference between the lengths of side BC.

Decomposition of Q_1:

- **Q**: Given that in triangle ABC, $\cos B = \frac{4}{5}$, $AC = b$, $AB = 10$, let $BC = a_1$, find the equation of these two variables. **A**: $b^2 = a_1^2 + 100 - 20a_1 \cdot \frac{4}{5} = a_1^2 - 16a_1 + 100$
- **Q**: Given that in triangle ABC, $\cos B = -\frac{4}{5}$, $AC = b$, $AB = 10$, let $BC = a_1$, find the equation of these two variables. **A**: $b^2 = a_2^2 + 100 - 20a_2 \cdot \left(-\frac{4}{5}\right) = a_2^2 + 16a_2 + 100$.

Contracted Question Q_2: Given that $b^2 = a_1^2 - 16a_1 + 100$, $b^2 = a_2^2 + 16a_2 + 100$, find the positive difference between a_1 and a_2.

Solution of Q_2:

Equating the two expressions for b^2:

$$a_1^2 - 16a_1 = a_2^2 + 16a_2,$$

and factoring:

$$(a_1 - a_2)(a_1 + a_2) = 16(a_1 + a_2).$$

Dividing by $a_1 + a_2$:

$$a_1 - a_2 = 16.$$

Thus, the positive difference between a_1 and a_2 is 16.

Final Answer: $\boxed{16}$.

D Implementation Details

D.1 Data Subset Selection

For the BBH dataset, we select all multiple-choice subsets to evaluate the model's logical reasoning capabilities. The selected subsets include temporal sequences, salient translation error detection, penguins in a table, snarks, ruin names, date understanding, hyperbaton, logical deduction (with three, five, and seven objects), movie recommendation, geometric shapes, disambiguation QA, and reasoning about colored objects. These subsets cover a diverse range of logical reasoning tasks, from temporal and spatial reasoning to deductive logic and error detection.

```
selected_sets = [
    'temporal_sequences',
    'salient_translation_error_detection
    ',
    'penguins_in_a_table',
    'snarks',
    'ruin_names',
    'date_understanding',
    'hyperbaton',
    'logical_deduction_five_objects',
    'movie_recommendation',
    'logical_deduction_three_objects',
    'geometric_shapes',
    'disambiguation_qa',
    'logical_deduction_seven_objects',
    'reasoning_about_colored_objects'
]
```

Listing 8: BBH Subset Selection

D.2 Forest of Thoughts

In our implementation, we utilize the classical Tree of Thoughts (ToT) approach as the fundamental tree structure in our Forest of Thoughts framework, while maintaining several critical mechanisms from the original FoT, including majority voting for aggregating results across different trees and expert evaluation for assessing solution quality. However, our implementation differs from the original FoT in certain aspects as we address a broader range of questions.

Specifically, we remove its early stopping criteria. The original FoT terminates tree splitting when nodes cannot produce valid outputs, which is particularly effective for mathematical question-solving like Game-of-24 where rule-based validation is straightforward. However, for our diverse use cases where output validity is less clearly defined, we maintain tree expansion regardless of intermediate output quality, allowing the framework to explore potentially valuable paths that might initially appear suboptimal. The Input Data Augmentation technique is also omitted since such analogical reasoning approach does not demonstrate consistent effectiveness across different types of questions.

These modifications allow the Forest of Thoughts framework to maintain the strengths of FoT while being more adaptable to a wider range of question domains. The implementation not only successfully reproduces the scaling curves reported in the original FoT paper but also achieves superior performance across multiple benchmarks.

D.3 AFlow

In our implementation, we leverage the optimal workflows identified by AFlow across different benchmark datasets while adapting them to suit our specific requirements. For mathematical reasoning tasks on MATH and GSM8k datasets, we directly adopt AFlow's established optimal workflows, which have demonstrated strong performance in these domains. Similarly, for multi-hop reasoning scenarios in LongBench, we utilize the workflow originally optimized for HotpotQA, as both datasets share fundamental multi-hop reasoning characteristics. For knowledge-intensive evaluation on MMLU-CF and logical reasoning tasks on BBH, which were not covered in the original AFlow paper, we conducted a new workflow search (consistent with the settings in the original paper) to identify the most effective approach, resulting in specialized workflows optimized for these formats.