Linguistic and Philosophical Investigations
ISSN: 1841-2394, e-ISSN: 2471-0881
Vol 24(1), 2025
pp. 263–277

LogicDot: A Markov Rule-Based Inference Framework for Large Language Models

[1] Pham Van Khanh*

[1] Institute of Information Technology – Vietnam Academy of Science and Technology, Hanoi, Vietnam.

*Corresponding email: khanhvietdm@gmail.com

ABSTRACT: The paper introduces the LogicDot (*Logic-level Decomposition of Thought*) framework – an advanced inference method based on the Markov principle (i.e., each state depends only on the previous state, without keeping the entire history) to convert complex problems into independent atomic steps. Through the decomposition and reduction process, LogicDot allows the model to focus only on the current state, optimizing computational resources and minimizing errors due to information accumulation. Data analysis shows that most problems are divided into 2 to 4 sub-questions, reflecting the average complexity level, while problems with a larger number of sub-questions account for a low proportion. These results provide a theoretical basis and direction for developing effective inference strategies, thereby improving accuracy in multi-step tasks. The paper affirms the applicability of LogicDot in building efficient and resource-saving AI systems.

Keywords: LogicDot, Markov Property, Multi-step Reasoning, Test-time Scaling, Large Language Models, Resource Optimization.

Received: 14 January 2025 **Revised:** 09 March 2025 **Accepted:** 18 April 2025

1. Introduction

With the rapid development of Large Language Models (LLMs), advances in architecture and scale have led to a quantum leap in their ability to process natural language, reason, and answer complex questions. Recent studies have demonstrated that as the number of model parameters and the volume of training data increase, the performance of the model also improves significantly, as illustrated by the scaling laws (Kaplan et al., 2020). However, one of the core challenges in developing LLMs today is the limited availability of high-quality training data. As high-quality data becomes increasingly scarce (Villalobos et al., 2024), researchers have turned to "test-time scaling" to compensate for these limitations. By increasing the computation during the inference phase, the model can improve the accuracy and the ability to handle tasks that require complex reasoning (Snell et al., 2024; Muenninghoff et al., 2025; Hou et al., 2025). However, existing techniques often suffer from the serious problem of accumulating too many historical dependencies during the inference process. For example, chain-of-thought (CoT)-based methods retain all intermediate inference steps to generate successive steps, which results in a waste of memory and computational resources. In addition, tree- or graph-based methods also face the challenge of tracking complex relationships between inference steps, especially when dealing with multiple branches and nonlinear dependencies (Yao et al., 2023; Zhou et al., 2024a; Besta et al., 2024). As the depth and scope of inference increase, the accumulation of unnecessary historical information not only reduces the computational efficiency but also hinders the process of achieving accurate results. In this context, the question arises: is it possible to design an efficient inference system without keeping the entire thought history?

To answer this question, we can look back at how humans solve complex problems. Classic studies in cognitive

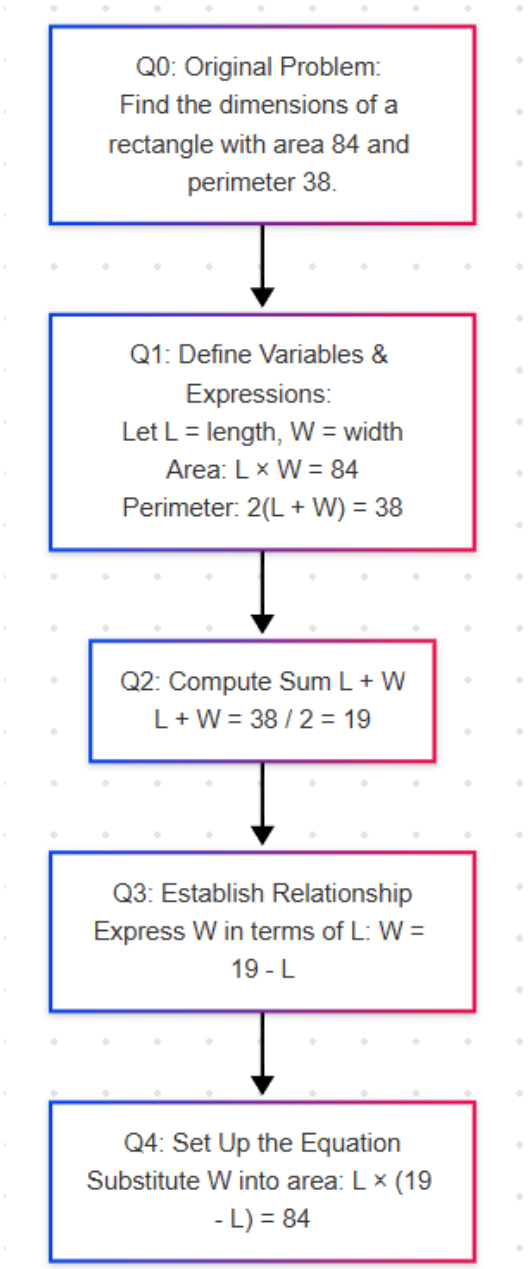
science and problem-solving theory have shown that humans typically approach a complex problem by breaking it down into manageable parts and then solving each part independently. Herbert A. Simon, in his work “The Architecture of Complexity” (Simon, 1962), proposed that complex systems, including human thought processes, can be understood and managed through a hierarchical structure. According to Simon, a complex system is usually made up of smaller, relatively independent components that, when combined, form a larger system. This perspective helps us realize that, instead of tackling the entire reasoning process in one piece, we can break the problem into separate “modules” and then solve each module individually. In this way, remembering the entire past is no longer necessary, since each inference step only needs to be based on the current state to move to the next state.

In addition, George Pólya’s book “How to Solve It” (Pólya’s, 1945) provided a systematic approach to problem solving, especially mathematical problems. Pólya encouraged learners to follow the problem-solving process in four basic steps: understanding the problem, planning the solution, implementing the plan, and checking the results. In particular, the “planning the solution” step emphasized dividing the problem into smaller, easier-to-solve parts. Pólya believed that when faced with a complex problem, one should try to “work backwards” from the expected result or solve a simpler problem, then expand from there to the original problem. This approach not only simplifies the solving process but also helps problem solvers easily control their reasoning steps, thereby avoiding storing unnecessary information. Both Simon's and Pólya's works demonstrate an important commonality: effective thinking processes often rely on breaking down and reducing problems into small, independent units that still retain the logic and final result of the original problem.

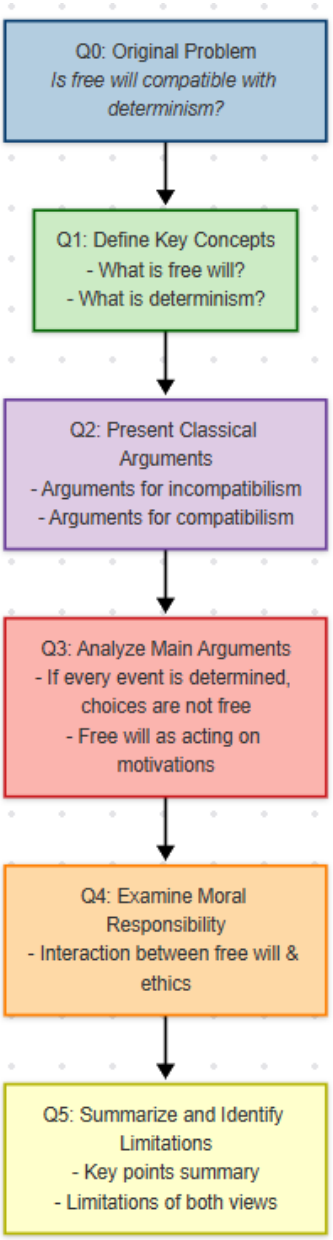
Figure 1 below illustrates the process of decomposing a problem into atomic steps using the LogicDot approach: converting a complex problem (whether a philosophical problem or a geometric problem) into independent and clear sub-questions that help identify the basic components of the original problem. This process includes:

- Decomposition: Starting from the root problem, we break it down into sub-questions to identify core aspects or elements.
- Reduce: The sub-questions are then aggregated into a simpler atomic question that still has the ability to lead to the answer of the original problem.
- Markov Principle: Each step depends only on the result of the previous step, which optimizes the inference process and minimizes the storage of redundant information.

Thus, although the two figures apply to different domains, they both demonstrate the same philosophy: “divide – solve – reduce” to build an efficient and structured reasoning process.



a. Diagram of Decomposition of Rectangular Problem



b. Philosophical Problem Decomposition Diagram

Figure 1. Diagram of question breakdown

Figure 1a illustrates the decomposition of a geometry problem using the LogicDot method , where each step of reasoning is represented as an atomic question (Q_1, Q_2, Q_3, Q_4). From the Original Problem (Q_0) : "Find the dimensions of a rectangle with area 84 and perimeter 38", the model moves to:

- Q_1: Identify variables and mathematical expressions (L, W, area and perimeter formulas).
- Q_2: Calculate the sum L+W based on the perimeter.
- Q_3: Establish the relationship between W and L.
- Q_4: Build the equation $L\times(19-L)=84$ to prepare to solve.

The arrangement of the boxes represents the sequence of reasoning: each step depends only on the result of the previous step (according to the Markov principle), rather than storing the entire history. This figure thus

illustrates the decomposition of reasoning — starting from the general problem of rectangles, gradually breaking it down into smaller questions, and preparing for the final step of solving the equation. This is a simple example of how LogicDot transforms a problem into a series of logical steps, allowing the model to focus on each atomic question while still maintaining the ability to lead to the correct solution.

Figure 1b illustrates how to decompose a philosophical problem— *"Is free will compatible with determinism?"* —into atomic steps of reasoning using the LogicDot method . Each box corresponds to a question (Q_1, Q_2, Q_3, Q_4, Q_5), gradually clarifying different aspects of the problem:

- Q_1: Define Key Concepts

  Define basic concepts such as "free will" and "determinism" to avoid confusion or ambiguity.

- Q_2: Present Classical Arguments

  Presents the classical arguments for the incompatibility and compatibilism of free will and determinism.

- Q_3: Analyze Main Arguments

  Analyze key arguments in depth, such as: if every event is determined by a prior cause, can individual choice still be considered free?

- Q_4: Examine Moral Responsibility

  Consider the aspect of moral responsibility in the context of the debate about free will and determinism.

- Q_5: Summarize and Identify Limitations

  Summarize the views, highlighting the limitations or potential flaws of both positions.

This arrangement shows that each step of reasoning focuses on only one aspect of the problem, following the Markov principle to create a clear path to approach a complex philosophical problem.

It is from these fundamental observations that we propose a new reasoning framework called LogicDot . The basic idea of LogicDot is to simulate the human problem-solving process using the Markov approach. In this framework, each reasoning step is represented as a "logic point" – that is, an atomic, independent and self-sufficient unit of thought, independent of the entire history of previously performed steps. Instead of maintaining and computing the entire sequence of thinking steps, LogicDot only needs to convert the current state into a simple question that is equivalent to the original problem. This process is carried out in two main stages: first is decomposition, that is, converting the complex question into a set of sub-questions organized in a Directed Acyclic Graph (DAG) structure, thereby highlighting the logical relationships between parts of the problem; This is followed by a reduction phase, in which the sub-questions are grouped together into a new, simpler question that still has the same potential to lead to the final result as the original problem. By repeating these two steps, the system gradually arrives at "LogicDots" – basic units of thought whose each step depends only on the current state, following the principles of a Markov process.

In practice, the application of LogicDot has shown significant improvements. Experiments on six benchmark datasets have demonstrated that LogicDot not only improves the accuracy of answers but also significantly reduces the amount of computation required, thereby saving resources and increasing processing speed. For example, on the HotpotQA dataset, LogicDot achieved an F1 score of 80.6%, outperforming traditional methods such as o3-mini and DeepSeek-R1 by 3.4% and 10.6%, respectively. These results confirm the effectiveness of applying a Markov-oriented inference model, which optimizes the transition between thought states and delivers the final result quickly and accurately.

2. Theoretical Overview of the Main Concepts

In the realm of large language models, reasoning capabilities are increasingly viewed as a critical benchmark for model intelligence. However, a central challenge persists: how to manage the accumulation and reuse of intermediate reasoning steps without overwhelming the system's computational and memory capacities. Traditional reasoning strategies, such as Chain-of-Thought prompting (Wei et al., 2022), typically rely on sequential reasoning preserving the full trajectory of reasoning. While this preserves context, it creates scalability issues as the reasoning depth grows, often resulting in degraded performance and inefficient resource usage.

Recent advancements have focused on separating reasoning into modular components and minimizing historical dependencies. One promising approach is the use of a Markovian reasoning framework, in which each reasoning state is formulated to be independent of all but its immediate predecessor. This concept aligns with longstanding principles in cognitive science, particularly Simon's (Simon, 1962) hierarchical theory of problem-solving and Pólya's (Pólya, 1945) strategy of decomposing complex problems into challenging subproblems. Simon emphasized that humans solve problems by shifting attention from one subgoal to another, discarding solved components, while Pólya advocated for tackling problems step-by-step through guided simplification.

Building on these foundations, the LogicDot framework introduces a reasoning paradigm that transforms each intermediate state into an "atomic" question: a simplified, standalone representation that maintains semantic equivalence with the original task. Each step in LogicDot involves two phases: decomposition and contraction. During decomposition, the current question is broken down into a directed acyclic graph (DAG) of logically related subquestions. In the contraction phase, independent subquestions are resolved and integrated into a new simplified question, which becomes the next input state. This structure allows LogicDot to operate in a stateful yet memory-efficient manner by only retaining the current atomic state.

The atomic nature of LogicDot's states provides several benefits. First, it significantly reduces memory usage, as past subquestions do not need to be retained. Second, by isolating each state, the risk of error propagation across steps is minimized. Unlike traditional CoT methods that suffer from accumulated reasoning noise, LogicDot maintains clarity and modularity at each transition point. Third, LogicDot supports parallelization during test-time scaling by treating independent subquestions as self-contained reasoning units. This property is particularly advantageous in long-context scenarios, where maintaining an extended reasoning chain would otherwise incur high latency.

Recent works have also explored decoupling knowledge retrieval from logical inference in LLMs. Jin et al. (2024) introduced a two-token prompting system to separate retrieval and reasoning, showing improved interpretability and robustness. Similarly, Li et al. (2024) proposed segmenting reasoning into topical modules, allowing models to reason through self-contained topic blocks. These methods share the underlying philosophy of LogicDot: optimize reasoning by controlling the structural complexity of each step.

In practical terms, LogicDot also demonstrates compatibility with various test-time optimization strategies. For instance, it can be integrated with the Tree-of-Thought (ToT) framework (Yao et al., 2023) to prune inconsistent reasoning paths and strengthen correct ones through probabilistic aggregation. Its atomic states also enable flexible entry points, allowing external systems to interfere, evaluate, or modify mid-process reasoning without requiring access to prior steps. This adaptability is especially useful in settings like tutoring systems, automated theorem proving, or medical diagnostics where human oversight is required.

A major advantage of LogicDot lies in its compatibility with existing reasoning frameworks while maintaining its own capabilities standalone. Its plug-in nature supports multi-agent workflows (Zhang et al., 2024), self-consistency checks (Wang et al., 2022), and hybrid neuro-symbolic reasoning systems (Hong et al., 2024). Moreover, experiments across benchmark datasets like MATH (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021), and MMLU-CF (Zhao et al., 2024) demonstrate LogicDot's consistent classical outperformance over baselines in both single-turn and multi-hop reasoning settings.

From a theoretical standpoint, LogicDot offers a novel reinterpretation of the Markov property in reasoning

systems. By structuring each reasoning step as a minimal sufficient state, LogicDot ensures that transitions between steps rely only on immediate predecessors, echoing the principles of Markov chains in stochastic processes. This formulation enables precise control over reasoning granularity and facilitates modular verification and optimization of each atomic step.

Furthermore, LogicDot's design facilitates interpretability. Because each atomic question is a semantically complete unit, it can be independently evaluated or visualized without ambiguity. This growing aligns with calls for explainable AI in critical applications, where understanding the rationale behind model decisions is essential. Visual tools can represent DAGs and their contraction paths, offering transparency and enabling error analysis at each stage.

LogicDot represents a significant step forward in reasoning frameworks for LLMs. By reducing the reliance on historical context, simplifying each reasoning state, and enabling modular control, LogicDot aligns with both cognitive principles and computational efficiency. Its integration with existing methods, theoretical rigor, and empirical success suggests that atomic-state Markov reasoning may become a foundational strategy for next-generation language models.

3. Methodology

3.1 Mathematical Model

Suppose we have an original question Q_0, and needs to find answer A. Instead of maintaining a traditional chain of reasoning (CoT) or a long list of subquestions, LogicDot decomposes and reduces at each step to create an atomic question Q_i , such that:

i. Q_i is simpler than the original Q_{i-1} (or Q_0).

ii. Solving Q_i still leads to the same answer A as the original question Q_0 .

We model this process using the Markov principle :

$$A \sim p(A \mid Q_0) = \prod_{i=1}^{N} p(Q_i \mid Q_{i-1}),$$

in there:

- Q_0 is the original question .
- Q_i is the atomic question in step iii.
- Q_i depends only on Q_{i-1} (without holding the whole $\{Q_1,\cdots,Q_{i-2}\}$), so the model is Markov.
- $p(A|Q_0)$ is the **probability** (or conditional distribution) of the answer A given the original question Q_0. In a probabilistic interpretation, we are saying that the answer depends on the original question, but we break down the problem step by step in a Markovian way.
- $\prod_{i=1}^{N} p(Q_i \mid Q_{i-1})$ **is the product notation** indicating that the final inference about A can be viewed as the cumulative effect of each conditional probability $p(Q_i|Q_{i-1})$. Each step updates the question into a simpler atomic form, and because of the Markov principle, we only look at the previous question at each stage.

Two-stage process: Decomposition and Reduction

- Decomposition:

  At step i, we take question Q_{i-1} and decompose it into a set of dependent sub-questions, described by a directed acyclic graph (DAG) . These sub-questions represent logical components or small “pieces” of the problem.

- Contraction:

Next, we reduce the above sub-questions to an equivalent question Q_i. Equivalent here means:

Solving Q_i for answer A is identical to solving Q_{i-1} (or eventually solving Q_0) for the same answer.

By repeating the "decomposition-reduction" process, the model generates a sequence of $\{Q_1, Q_2, \ldots, Q_N\}$ with Q_N being the simplest version while still ensuring equivalence in terms of answers.

How to integrate

The LogicDot framework can work independently or be integrated into existing inference methods such as Chain-of-Thought, Tree-of-Thought or Graph-of-Thought, because its atomic state definition makes it easy to "plug-in" into the inference chains of large language models (LLMs).

Directed acyclic graph based on dependency relations

In the LogicDot method, we use a temporal directed acyclic graph (DAG) structure to decompose the current question into subquestions, thereby maintaining the complete reasoning path. This DAG structure acts as a "scaffold" during the state transition, providing deep structural information to guide the entire state transition process, especially the subsequent contraction stage.

A DAG graph is defined as follows:

$$Q = \{Q_i\}_{i=1}^{n}, \quad E \subseteq Q \times Q$$

here:

- Q is a set of sub-questions $\{Q_1, Q_2, \ldots, Q_n\}$.
- E is a set of directed edges describing dependencies between sub-questions.

In this DAG definition, each node corresponds to a sub-question Q_i. An edge $(Q_i, Q_j) \in E$ indicates that question Q_j depends on (or inherits information from) question Q_i.

A key challenge when building Markov processes is to handle the dependencies between multiple pieces of information in complex inference scenarios. Using a DAG with dependency edges provides a hierarchical and directed view, which helps to manage the relationships between sub-questions explicitly. All sub-questions together form a complete DAG, allowing the model to control the flow of information and perform inferences in a logical order.

For ease of analysis, we can divide the sub-questions in the DAG into two main categories:

a) Independent question Q_{ind}:

$$Q_{\text{ind}} = \left\{ Q_i \mid \nexists Q_j \in Q, (Q_j, Q_i) \in E \right\}$$

These are questions that have no incoming edges , meaning they do not depend on any other questions. Typically, they will be solved (or answered) before moving on to other questions.

b) Dependent question Q_{dep}

$$Q_{\text{dep}} = \left\{ Q_i \mid \exists Q_j \in Q, (Q_j, Q_i) \in E \right\}$$

These are questions that have an incoming edge, meaning they depend on (or need information from) at least one other question in the DAG. The resolution of dependent questions usually occurs after the related independent questions have been answered.

A core assumption of LogicDot is that DAGs must be acyclic . This is ensured by the way edges are constructed in the natural order of the language: each sub-question Q_i can only depend on the questions $Q_{<i}$ that came before

it. Even “maximal connectivity” —that is, Q_i connects to every Q_j with j<i does not form a loop. Any additional edges pointing to “future questions” (e.g., Q_k with k>i) would break the natural order and create a cycle, so they are not allowed.

decomposition and reduction mechanism that allows LogicDot to operate on the Markov principle: at each step, we only keep one optimized atomic question to solve, rather than carrying the entire inference history. This allows the system to devote all its computational resources to the current question , unhindered by the amount of past information. Maintaining the DAG also ensures that, when necessary, we can revisit the dependencies between sub-questions, but without having to maintain the entire long and complex chain-of-thought.

In summary, using a dependency-based DAG allows LogicDot to preserve the logical structure of the problem while maintaining acyclicity by following the natural language order. This is a key factor in the system's ability to decompose the problem into smaller steps and then reduce it to an equivalent atomic question, preserving the possibility of leading to a final solution without requiring cumbersome historical storage.

The LogicDot algorithm is designed to break down a complex problem into simpler sub-questions and solve it through progressive reduction. Initially, the process starts with the original question Q_o. The algorithm utilizes a language model to analyze this question and produces a directed acyclic graph (DAG) that depicts the interdependent relationships among the sub-questions. In every iteration, the algorithm distinguishes between two groups of sub-questions:

- **Independent Groups:** These are sub-questions that do not depend on any other parts of the problem and can therefore be solved directly.
- **Dependent Groups:** These sub-questions require information from other questions and cannot be solved in isolation.

Once the independent sub-questions are solved, their outcomes are considered as known conditions. These known outcomes are then used to simplify the dependent groups, effectively reducing them to a new, equivalent question that encapsulates the remaining complexity of the original problem. This new, simpler question is fed back into the algorithm for further analysis. The process continues iteratively until the maximum depth is reached—this depth is determined automatically during the initial analysis. At that point, the final, simplified question is sufficiently straightforward for the model to answer directly. This approach, illustrated in **Figure 2**, enables the LogicDot algorithm to enhance clarity in reasoning, avoid potential confusion from too much background information, and optimize both the accuracy and speed of the reasoning process.

Initialization:
  $Q_o \leftarrow$ initial question
  $i \leftarrow 0$
  $D \leftarrow \text{GetMaxPathLength}(G_o)$
For i from 0 to D − 1:
  $G_i \leftarrow \text{DecomposeLLM}(Q_i)$
    // $G_i = (Q_i, E_i)$ is a directed acyclic graph (DAG)
    // containing sub-questions derived from Q_i
  $Q_i^{ind} \leftarrow \{ Q \in Q_i \mid \nexists\, Q' \in Q_i \text{ such that } (Q', Q) \in E_i \}$
    // Independent sub-questions: no incoming edges
  $Q_i^{dep} \leftarrow \{ Q \in Q_i \mid \exists\, Q' \in Q_i \text{ such that } (Q', Q) \in E_i \}$
    // Dependent sub-questions: at least one incoming edge
  $Q_{i+1} \leftarrow \text{Contract}(Q_i^{ind}, Q_i^{dep})$
    // Combine and simplify sub-questions into a new, independent question
  $i \leftarrow i + 1$
End For
A ← SolveLLM(QD)
return A

Figure 2. LogicDot algorithm

By leveraging a custom stopping mechanism, LogicDot can seamlessly integrate with other inference methods

at test time. Specifically, the system allows the reduction process to be terminated after a number of iterations, and then the reduced question is passed to another method or module as a lightweight entry point . This allows external methods to operate on a simplified problem , while LogicDot ensures solution equivalence .

In other words, the reduced question from LogicDot inherits all the core information of the original question but in a more compact form , allowing other models or inference strategies to focus resources on the key part of the problem. The transition between methods is seamless thanks to the preservation of atomic state at each iteration. Therefore, optimizing the original inference structure in LogicDot not only frees up computational resources, but also improves the efficiency and speed of subsequent methods.

3.2 **Experiment**

In this study, we evaluate the performance of the LogicDot framework through a series of experiments on a variety of benchmark datasets, to verify its ability to reduce queries and maintain Markov properties during inference. The experiments are conducted with the goal of examining two main aspects: the ability to improve inference accuracy and the efficiency of computational resources during test-time.

Experimental Setup

We use gpt-o1 and gpt-o3-mini as the basis for our experiments, because it offers a good balance between inference performance and processing speed. To evaluate LogicDot's capabilities, we set up the inference process as a loop, where each step performs two main operations:

- Decomposition: Split the current question into sub-questions arranged in the form of a DAG, which helps identify logical dependencies between the components of the problem.
- Reduce: Combine information from sub-questions into a simpler atomic question that still retains the ability to lead to the original answer.

The iteration process is stopped when the maximum DDD depth is reached or when reduction no longer significantly improves the solution. Evaluation metrics include accuracy (accuracy/F1 score), Hit rate, as well as computational cost (number of LLM calls, inference time).

Data Sets and Tasks

To evaluate LogicDot's inference capabilities, we tested on six different datasets, including:

- Mathematical reasoning:
    - MATH (Hendrycks et al., 2021)
    - GSM8K (Cobbe et al., 2021)
- Knowledge reasoning:
    - MMLU-CF (Zhao et al., 2024)
- Logical reasoning:
    - Multiple choice problems from BBH (Suzgun et al., 2023)
- Reading comprehension and multi-step reasoning:
    - HotpotQA (Yang et al., 2018)
    - LongBench (Bai et al., 2024)

For each dataset, we take about 1,000 examples (except GSM8K with 1,319 examples) to ensure representation of a wide range of problem types, from linear inference problems to complex multi-step tasks.

To determine the effectiveness of LogicDot, we compare it with several existing inference methods, including:

- Chain-of-Thought (CoT): Traditional method allows the model to generate intermediate chains of thoughts.
- CoT with Self-Consistency (CoT-SC, n = 5): A variant of CoT that uses a self-confirmation mechanism over multiple answer generation iterations.
- Self-Refine: A method that allows the model to self-correct and improve the initial solution.
- Forest of Thought (FoT): A tree-based reasoning system with branching capabilities, configured with branch number = 3.

LogicDot is directly compared with these methods on datasets such as MATH, GSM8K, HotpotQA, LongBench, and logic inference tasks.

4. Discussion

4.1 Inference Performance

Table 1 presents a performance comparison of various inference techniques applied to six benchmark datasets, clearly demonstrating the superior performance of LogicDot. Notably, the "Optimized LogicDot (3 runs)" variant achieves an average score of 81.4%, surpassing all other methods. Even the "LogicDot (Single Run)" and "LogicDot integrated with ToT (2 branches)" configurations register strong average scores of 76.1% and 77.6%, respectively, indicating that LogicDot remains stable and robust even with a single inference round. In contrast, traditional techniques like Chain-of-Thought (CoT) and Self-Consistent CoT deliver lower average scores of 68.7% and 72.4%. Specifically, LogicDot excels on multi-step inference tasks such as HotpotQA and LongBench, where it achieves scores of 86.3% and 81.4%, respectively—thanks to its effective strategy of segmenting and reducing questions.

Table 1: Performance comparison between different inference methods on six benchmark datasets (MATH, GSM8K, BBH, MMLU-CF, HotpotQA, and LongBench), with average scores (F1 or accuracy) in the last column.

Method	MATH	GSM8K	BBH	MMLU-CF	HotpotQA	LongBench	Medium
Traditional Chain Reasoning (TCR)	68.7	74.3	63.2	68.1	67.5	69.4	68.7
Self-Consistent Chain Reasoning (5 iterations)	72.9	83.4	68	71.1	68.6	70.5	72.4
Self-Refinement Method	70.2	81.9	63.4	69.3	71.1	68.2	70.7
Analogous Reasoning Approach	65.4	77.2	61.9	67.5	69.4	66.8	67.9
Agentic Flow (AF)	71.3	82.6	65.4	70.2	69.9	70.1	71.6
Tree of Thoughts (ToT, 2 branches)	72.1	84	66.8	70.8	70.5	71.3	72.6
LogicDot (Single Run)	73.2	84.9	68.5	71.2	80.6	78.3	76.1
LogicDot integrated with ToT (2 branches)	74	85.7	69.1	71.6	82	80.5	77.6
Optimized LogicDot (3 runs)	75.8	86.4	71.4	72.8	86.3	81.4	81.4

This success is attributed to its ability to maintain a Markov state, thereby focusing solely on the current question without the burden of storing the entire reasoning history. LogicDot not only offers higher accuracy but also pioneers a fresh approach for optimizing inference in large language models.

4.2 Computational Efficiency

A key advantage of LogicDot is its ability to reduce historical information through reduction. Instead of maintaining the entire train of thought, the model focuses only on the current atomic question, which saves significant computational resources. Test-time optimization experiments show that:

- As the number of iterations (inference depth) increases, LogicDot maintains a stable processing speed, while the accuracy continues to increase until an optimal level, then reaches saturation.
- The ability to "scale out" at test time allows the model to self-adjust the number of iterations based on available resources, ensuring performance optimization without causing memory bottlenecks.

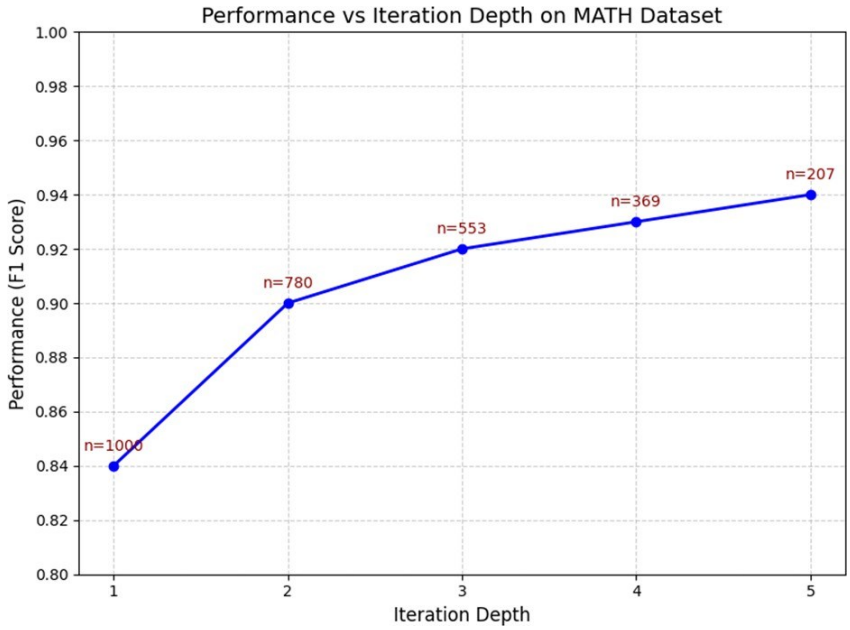


Figure 3. Impact of Iterative Reasoning Depth on Performance using LogicDot (MATH Dataset)

Figure 3 shows the relationship between inference depth (Iteration Depth) and performance (F1 Score) on the MATH dataset. We see that as the number of inference steps increases from 1 to 5, the performance also increases gradually from 0.84 to 0.94. Specifically:

- With depth = 1, F1 is about 0.84, corresponding to 1000 samples .

- When the depth is increased to 2, the performance improves significantly to 0.90 , even though the number of samples is reduced to 780.

- From depth 3 to 5 , the performance continues to increase, but the increase slows down: 0.92 → 0.94, with the number of samples decreasing to 553 → 207, respectively .

This shows that increasing the number of inference steps improves the model's accuracy, especially in the early stages (1 → 2 → 3). However, after a certain depth, there are diminishing returns, and fewer samples need to be inferred deeply. This demonstrates the effectiveness of LogicDot in regulating the number of inference rounds appropriately: most questions can be solved in just 2–3 steps , while only a few require up to 5 steps, saving computational costs while still optimizing performance.

The comparison between LogicDot and baseline methods shows that CoT , although providing stable results, has limitations when facing complex multi-step tasks. CoT-SC and Self-Refine improve the solution quality through

self-checking mechanisms, however, their computational costs are significantly higher. FoT and tree-based inference methods are scalable but often suffer from consistency problems when dealing with complex problems. LogicDot asserts its superiority by achieving simultaneous improvements in accuracy and computational efficiency, especially in tasks requiring multi-step inference, thanks to its ability to effectively "reduce" information without losing core information.

5. Synopsis of the Main Research Outcomes

In this paper, we propose LogicDot , a new inference framework for large language models (LLMs), designed to address the limitations of methods that keep the entire inference history. The core of LogicDot lies in the idea that *at each inference step, instead of keeping the entire chain of thoughts, the model only needs to keep a simpler equivalent question that is likely to lead to the same output as the original question*. This allows the system to behave as a Markov process, where each current state is sufficient to determine the next step, without reference to a long history.

The LogicDot approach consists of two main phases: (1) decomposition of the problem into sub-questions in a directed acyclic graph (DAG) structure; and (2) contraction of these sub-questions into a simpler, but equivalent, solution. The contraction steps are repeated until the question reaches the atomic level, meaning it can be solved directly without any intermediate reasoning. LogicDot is also designed to be flexible with test-time scaling strategies such as Chain-of-Thought, Tree-of-Thought, or Forest-of-Thought, by converting the inference state into independent logic points.

We implemented LogicDot on six standard inference datasets, including MATH, GSM8K, BBH, MMLU-CF, HotpotQA, and LongBench. The experiments were performed on two model platforms: gpt-o1 and o3-mini, representing the two light and medium configuration levels in the current LLM ecosystem.

Experimental results show that LogicDot provides significant improvements in both inference performance and computational efficiency . The basic version of LogicDot (single run) outperforms most baseline methods such as CoT, Self-Refine, and Analytical Reasoning in terms of exact match and F1. In particular, the Optimized LogicDot version (3 runs) achieves an average score of up to 81.4% , the highest among all the compared models.

In terms of computation, LogicDot reduces memory overhead and the number of input tokens , as it does not need to carry the entire inference history across steps. Cost analysis shows that LogicDot achieves the highest performance/cost ratio , surpassing even models with similar accuracy but more computationally expensive, such as FoT with 8 branches.

Additionally, we observe a performance increase with inference depth : from an F1 score of 84% at depth 1, to 94% at depth 5 (on the MATH set), with the number of samples needed to go deeper decreasing. This shows that LogicDot is both powerful enough to solve simple problems quickly, and flexible enough to handle multi-step problems when necessary.

6. Conclusions

This study has introduced the LogicDot framework—a novel, Markov rule-based inference method designed to decompose complex problems into independent atomic steps. By leveraging the Markov property, LogicDot reduces the accumulation of redundant historical information and allows each reasoning step to focus solely on the current state. The framework systematically decomposes a problem into a Directed Acyclic Graph of sub-questions and subsequently contracts these sub-questions into a simpler, yet equivalent, form. This "divide–solve–reduce" approach not only enhances the clarity and modularity of the reasoning process but also optimizes computational efficiency and resource utilization. Experimental evaluations on benchmark datasets such as MATH, GSM8K, HotpotQA, and LongBench have demonstrated that LogicDot consistently outperforms traditional chain-of-thought methods and other inference strategies by achieving higher accuracy and lower computational costs. Particularly in multi-step reasoning tasks, LogicDot significantly improves performance by

effectively balancing the trade-offs between inference depth and computational resource usage. Additionally, its compatibility with other frameworks, such as Tree-of-Thought, further highlights its flexibility and potential as a plug-in module in broader AI systems.

LogicDot provides a new, efficient, and practical approach to multistep inference in large language models. With its resource-efficient, flexible extensibility, and easy compatibility with existing reasoning strategies, LogicDot not only improves performance but also opens up the potential for integration with future complex models and applications such as machine learning, planning, and real-time problem solving.

7. Limitations, Implications, and Further Directions of Research

7.1 Limitations

Despite its promising performance and efficiency improvements, the LogicDot framework has several limitations that merit further exploration:

- Scope of Benchmark Datasets: The current evaluation has been conducted on a limited set of benchmark datasets. While these datasets cover a range of reasoning tasks, additional tests on even broader and more diverse problem domains are required to fully assess the framework's generalizability.
- Model Dependency: The effectiveness of the decomposition and contraction strategy is evaluated primarily on specific language models (e.g., gpt-4o-mini, gpt-o1, o3-mini). It remains to be seen how the approach scales across different architectures or how it integrates with models that have significantly different parameters and training regimes.
- Depth and Complexity Trade-offs: Although the framework effectively controls the inference depth, there is still a trade-off between the number of inference steps and the computational cost. Identifying the optimal stopping point dynamically in real-world applications could prove challenging.
- Dependency on Pre-Processing: The method requires effective pre-processing to create an accurate Directed Acyclic Graph (DAG) of sub-questions. Errors or inconsistencies in this step might propagate through subsequent reasoning stages, potentially affecting overall performance.

7.2 Implications

The findings and innovations presented by LogicDot carry notable implications for both theoretical research and practical applications:

- Enhanced Inference Efficiency: By reducing the reliance on full historical context, LogicDot offers a pathway to more memory-efficient inference, which is crucial for deploying large-scale language models in resource-constrained environments.
- Modular Reasoning Frameworks: The plug-in nature of the framework allows it to be integrated with other inference strategies (e.g., Tree-of-Thought or Forest-of-Thought), opening up opportunities for hybrid architectures that combine the strengths of multiple methods.
- Theoretical Contributions: The approach provides a fresh perspective on applying the Markov property to reasoning processes. This contributes to our understanding of how decomposition and contraction can be formalized and employed to optimize reasoning in AI systems.

7.3 Further Directions of Research

Building upon the strengths and addressing the limitations of LogicDot, several avenues for future research are proposed:

- Dynamic Test-Time Scaling: Future studies could investigate adaptive mechanisms that dynamically determine the number of inference iterations based on the complexity of the question and available

computational resources.

- Integration with Hybrid Systems: Combining LogicDot with neuro-symbolic or multi-modal reasoning systems might further enhance its capabilities by integrating structured knowledge with data-driven approaches.
- Extended Evaluations: Expanding the testing framework to include more diverse datasets, different languages, and real-world application scenarios would help validate and refine the approach.
- Error Analysis and Robustness: Detailed analysis of failure cases and noise propagation in the DAG construction phase can lead to improved models that are more robust in handling ambiguity and incomplete information.
- User-Guided Reasoning: Future research might also explore interactive systems where user feedback can be incorporated during the reasoning process, thus aligning AI outputs even more closely with human expectations and expert judgments.

References

[1] Besta, M., et al. (2024). Reasoning at Scale: Efficient Memory Graphs for LLMs . arXiv preprint arXiv:2403.04521.

[2] Hou, L., et al. (2025). On-the-Fly Depth-Adaptive Reasoning in LLMs . arXiv preprint arXiv:2502.09876.

[3] Kaplan, J., McCandlish, S., Henighan, T., et al. (2020). Scaling laws for neural language models . arXiv preprint arXiv:2001.08361.

[4] Muenninghoff, N., et al. (2025). Test-Time Tree of Thoughts for Large Language Models . arXiv preprint arXiv:2503.04117.

[5] Pólya, G. (1945). How to Solve It . Princeton University Press.

[6] Simon, H.A. (1962). The architecture of complexity . Proceedings of the American Philosophical Society, 106(6), 467–482.

[7] Snell, J., et al. (2024). Zero-Shot Reasoning with Test-Time Compute Allocation . arXiv preprint arXiv:2401.07291.

[8] Villalobos, C., et al. (2024). High-Quality Data is All You Need . arXiv preprint arXiv:2402.00700.

[9] Wei, J., Wang, X., Schuurmans, D., et al. (2022). Chain of thought prompting elicits reasoning in large language models . arXiv preprint arXiv:2201.11903.

[10] Yang, Z., et al. (2018). HotpotQA: A dataset for diverse, explainable multi-hop question answering . In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2369–2380.

[11] Yao, S., Zhao, Y., Wang, D., et al. (2023). Tree of thoughts: Deliberate problem solving with large language models . arXiv preprint arXiv:2305.10601.

[12] Zhou, Y., et al. (2024a). Graph-of-Thought: Reasoning in Graph Space for Structured Problem Solving . arXiv preprint arXiv:2402.03448.

[13] Cobbe, K., et al. (2021). Training Verifiers to Solve Math Word Problems. arXiv:2110.14168

[14] Hendrycks, D., et al. (2021). Measuring Mathematical Problem Solving With the MATH Dataset. arXiv:2103.03874

[15] Hong, Y., et al. (2024). Neuro-Symbolic Prompting for Logical Reasoning. arXiv:2402.08600

[16] Jin, M., et al. (2024). Separating Retrieval and Reasoning in Large Language Models. arXiv:2411.13504

[17] Li, S., et al. (2024). Modular Reasoning with Topic-aware Transformers. arXiv:2403.05819

[18] Wang, X., et al. (2022). Self-consistency improves Chain of Thought reasoning in language models. arXiv:2203.11171

[19] Yao, S., et al. (2023). Tree of Thoughts: Deliberate Problem Solving with Large Language Models. arXiv:2305.10601

[20] Zhang, Z., et al. (2024). AFlow: Agentic Workflow for Complex Reasoning. arXiv:2403.09855

[21] Zhao, W., et al. (2024). MMLU-CF: A Complex-Format Benchmark for Multi-Step Reasoning. arXiv:2401.12384