Beyond Translation: A Decomposed Collaborative Reasoning Framework Harnessing LLMs and Symbolic Solvers

Zhixin Zhang[1*] **Shaobo Zhang**[1*] **Chengcan Wu**[1*] **Meng Sun**[1†]

[1]Peking University

Abstract

The integration of large language models (LLMs) with symbolic solvers represents a promising direction for enhancing logical reasoning capabilities. However, existing methods often underutilize LLMs, treating them primarily as text-to-symbol converters, and remain prone to hallucinations and performance degradation in complex reasoning scenarios requiring multi-step intermediate reasoning. To address these limitations, we propose a novel framework that more deeply integrates LLMs with logical solvers. Our approach employs a two-step, fine-grained decomposition of the reasoning process. First, the LLM analyzes the problem and provides an outline for solving it. Second, the LLM decomposes the original problem into simpler sub-problems and generates formal conditions for each. This strategy fully leverages the LLM's strengths in contextual exploration and problem structuring while offloading rigorous deduction to the symbolic solver, thereby mitigating hallucinations. Extensive experimental evaluation demonstrates that our framework significantly outperforms state-of-the-art baselines. Our work provides a more effective paradigm for applying LLMs to complex reasoning tasks. Our codes are available at https://github.com/meloxxxxxx/DCRF

Introduction

The integration of logical reasoning has long been a significant challenge in the field of Artificial Intelligence and Natural Language Processing (Newell and Simon 1956; McCarthy and Hayes 1981). The application of reasoning spans multiple fields (Manning 2022) such as problem solving, theorem proving, decision making, and robotics, and is crucial for the development of artificial intelligence. In recent years, with the rapid advancement of large language models (LLMs) (Achiam et al. 2023; Touvron et al. 2023; Liu et al. 2024), it has been observed that these models begin to exhibit certain reasoning capabilities (Wei et al. 2022a). When employing the Chain-of-Thought (CoT) technique, LLMs are instructed to reason step-by-step (Wei et al. 2022b; Kojima et al. 2022), in some cases even demonstrating reasoning abilities akin to humans (Dunivin 2024). Despite substantial progress in reasoning capabilities, Large Language Models (LLMs) still exhibit several deficiencies in solving reasoning problems (Liu et al. 2023): several studies have found (Golovneva et al. 2022; Ribeiro et al. 2023; Lyu et al. 2023) that LLMs occasionally demonstrate reasoning biases, leading to conclusions that are logically inconsistent with previously generated reasoning chains; other research indicates that current language models still lack robust reasoning abilities, showing unstable performance across different datasets (Xu et al. 2025); furthermore, the hallucination problem in language models can also lead to reasoning errors (Chen and Shu 2023; Li et al. 2023; Mündler et al. 2023), particularly when simple semantic coherence is insufficient for more complex logical reasoning domains.

To address these limitations, recent research trends have focused on a neuro-symbolic framework that integrates language models with symbolic solvers to enhance performance, including LogicLM (Pan et al. 2023), LINC (Olausson et al. 2023), CLOVER (Ryu et al. 2024), and QuaSAR (Ranaldi, Valentino, and Freitas 2025). These approaches primarily leverage language models as text-to-symbol converters, relying on traditional external reasoning systems for the core inference process.

Although they ensure rigorous answer verification using logic solvers, they underperform on complex logical datasets, particularly those requiring detailed explanations of intermediate reasoning steps. This stems from the insufficient utilization of LLMs within their reasoning workflow. On one hand, they overlook the LLM's ability to capture key aspects of the problem statement through planning and deliberation, leading to suboptimal translations. On the other hand, using the logic solver alone for reasoning fails to generate interpretable solutions; thus, LLMs need to be involved in the reasoning process itself, not merely as translators. These shortcomings result in a significant gap between their practical applicability and the requirements for solving complex logical problems (Drori et al. 2022).

Other works, like Symbolic CoT (Xu et al. 2024), attempt to more tightly integrate the CoT technique of LLMs with logical languages, aiming to better leverage the reasoning capacity of LLMs and improve overall reasoning effectiveness. These methods behave well in explicable deductions, but the hallucination issues remain unsolved, as it does not employ rigid logic solvers for verification.

To address the aforementioned challenges and enhance

*Equal contribution.

†Corresponding to Meng Sun (sunm@pku.edu.cn).

AAAI 2026 Bridge Program on Logical and Symbolic Reasoning in Language Models (LMReasoning).

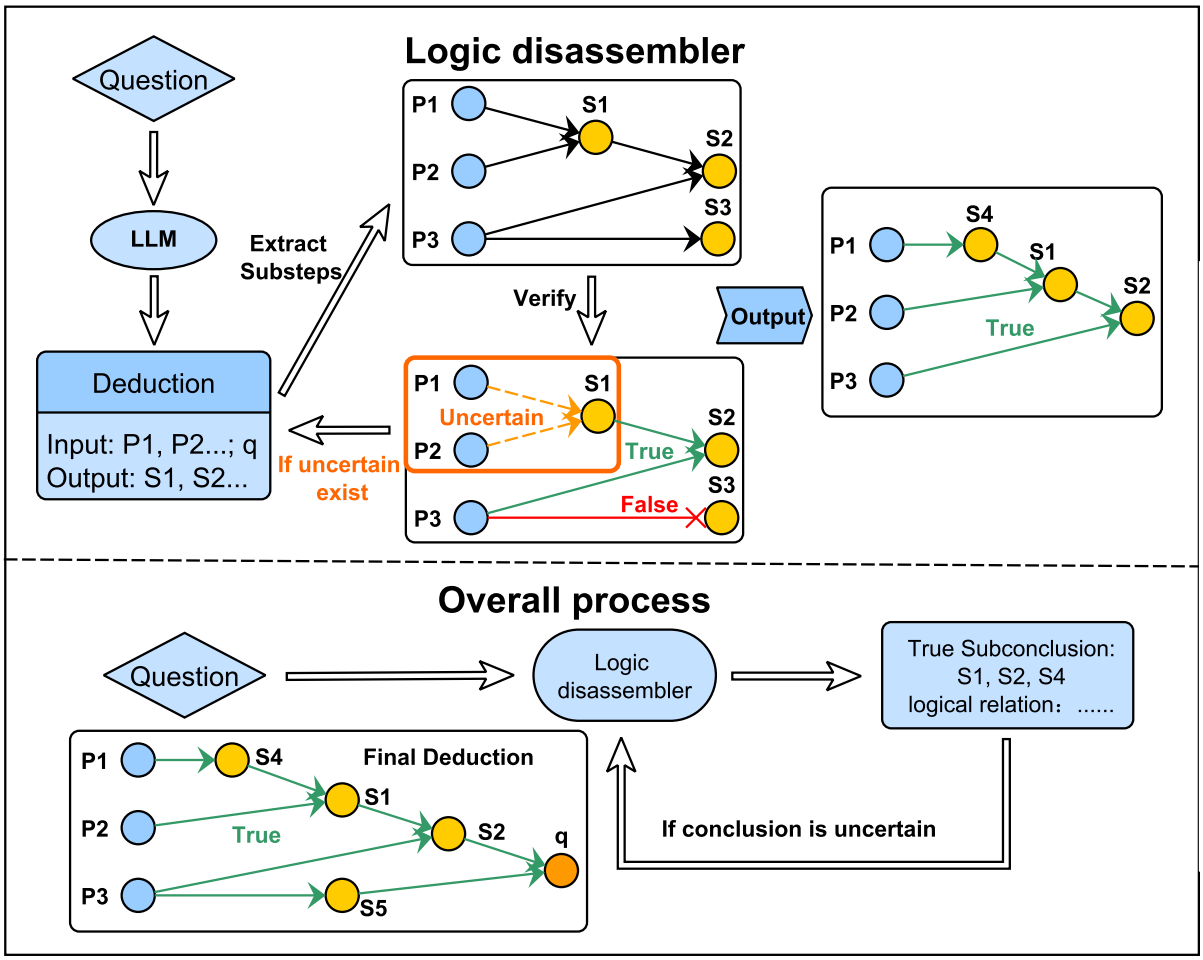


Figure 1: Method overview. We disassemble the original problem into simpler substeps, *i.e.* substeps $S1, S2, \ldots$ and an answer q, by extracting them from the LLM deduction. Then we verify and solve the acquired substeps using the logic solver. If the substep carries unverifiable skipping of steps, *i.e.*, the logic solver returns $Uncertain$, we continue to decompose it for further verification recursively, until we reach a complete and verifiable deduction graph. During the whole process, we correct the deduction steps based on the verification results and iterate to reach an optimal deduction. Our framework is formally summarized in Algorithm 1.

the performance of integrated LLMs and logical solvers on complex reasoning tasks, we propose a novel methodological framework. The core intuition behind our framework is to leverage the deduction capabilities of LLMs more comprehensively while mitigating their reliability limitations in complex logical reasoning. Specifically, we recursively decompose the problem into simple atomic deduction steps, then verify and solve the acquired atomic steps using the logic solver. This framework ensures that the LLM has an adequate understanding of the problem through decomposition, thus capturing crucial points in the problem statement for correct LLM translation and logic solver verification. Moreover, the acquired deduction steps with logic solver verification generate a complete and rigid deduction process, ensuring strong interpretability compared to prior neuron-symbolic methods. The process above iterates under the guidance of the logic solver verification until we reach a rigid conclusion. A demonstration of our framework is shown in Figure 1.

Our method makes the LLM transcend the role of a mere translator, as we more fully harness its capabilities for contextual exploration and problem structuring, areas where LLMs naturally excel. Correspondingly, we circumvent the reliability issue of LLMs in complex logical deduction, preventing hallucination behaviors. Extensive experimental evaluation demonstrates that our method achieves superior performance across multiple logical reasoning benchmarks, with an average accuracy improvement of 15% over most baseline approaches. Moreover, our method excels other methods by generating a complete and rigid deduction process, substantiating both its reasoning precision and practical effectiveness for complex logical reasoning tasks.

This work makes the following key contributions:

- We critically re-examine the limitations prevalent in current neuron-symbolic frameworks, identifying their major shortcoming: the absence of LLMs in their deduction,

which results in both suboptimal problem translation and the lack of interpretable deductions.

- We propose a novel LLM-solver integration framework that strategically leverages the deduction capability of LLMs while systematically mitigating hallucination risks with logic solvers, thereby enabling effective reasoning in practical applications.
- Through comprehensive experiments, we demonstrate the significant effectiveness of our framework, showing consistent outperformance over existing baselines across diverse logical reasoning datasets, providing valuable insights and advancements for applying LLMs to complex logical reasoning challenges.

Related Work

Logic Solver: The long-standing pursuit of automated logical reasoning has been a central goal of artificial intelligence, leading to the development of various solvers with distinct methodologies. Traditionally, this field has been dominated by symbolic reasoning systems, such as Automated Theorem Provers (e.g., Prover9 (McCune 2005), Pyke (Frederiksen 2008)) and Satisfiability Modulo Theories (SMT) solvers (e.g., Z3 (De Moura and Bjørner 2008)). These systems operate based on formal representations like first-order logic, employing rigorous deductive rules to derive conclusions with guaranteed logical soundness. Their primary strengths lie in reliability and transparency; however, they are often brittle and require perfectly structured inputs, rendering them inaccessible for problems expressed in raw natural language.

Chain of Thought: With the advancement of language models, reasoning capabilities have garnered significant attention. (Wei et al. 2022b) proposed Chain of Thought (CoT) prompting, which guides LLMs to generate responses in a structured 'input, reasoning, output' format, where the "reasoning" component exhibits coherent, step-by-step natural language reasoning leading to the final answer. These intermediate steps prompt the model to reason progressively through different logics, enhancing its ability to solve complex problems, such as arithmetic, symbolic reasoning, and logical reasoning (Sprague et al. 2024; Luo et al. 2025; Zhao et al. 2023). Although CoT techniques can enhance the reasoning capabilities of LLMs, they struggle to eliminate hallucinations during reasoning (Akbar et al. 2024) and still suffer from a lack of interpretability (Barez et al. 2025). This makes it difficult to fine-tune and correct errors when mistakes occur in logical reasoning tasks. In summary, existing CoT techniques still face challenges in enabling LLMs to perform robust reasoning in complex environments (Sprague et al. 2023).

LLM Logical Reasoning: Logic-LM (Pan et al. 2023) first proposed integrating LLMs with logical solvers by using the LLM to translate natural language into formal logic representations, which are then processed by the solver to enhance reasoning. However, the limited translation capability of language models resulted in suboptimal performance. Subsequent research proposed various methods to improve translation accuracy. LINIC (Olausson et al. 2023) suggested having the language model translate K times followed by voting to increase accuracy. CLOVER (Ryu et al. 2024) decomposed natural language logic problems into potential logical components in first-order logic forms and selected the most probable logical combination, thereby improving semantic accuracy. QuaSAR (Ranaldi, Valentino, and Freitas 2025) enhanced semantic logic by having the LLM identify key logical components in natural language problems and transform them into quasi-formal representations—a semi-structured mix of symbols and natural language. However, these methods primarily focus on the translation capability of language models, without leveraging other LLM abilities such as searching, association, or simple reasoning. Consequently, SymbCoT (Xu et al. 2024) proposed a framework relying solely on language models, dividing the LLM into four modules—translation, planning, solving, and verification to perform multi-step reasoning centered around symbolic logic. This method performs well in simple reasoning tasks, but the hallucination issues of language models become apparent in complex reasoning scenarios, as it does not employ rigid logic solvers for verification. As language models continue to evolve, the future direction should not be confined solely to optimizing translation capabilities nor simply assigning all tasks to the LLM. Instead, it is crucial to properly acknowledge and integrate the various strengths and weaknesses of LLMs, better leveraging their advantageous capabilities while avoiding their limitations, such as hallucination in complex reasoning.

Method

Motivation

LLMs suffer from reliability issues in complex reasoning (Chen and Shu 2023; Li et al. 2023; Mündler et al. 2023), including hallucination, cyclic reflection, lack of verifiability and faithfulness, *etc.* To mitigate these reasoning flaws, prior works, including LogicLM (Pan et al. 2023), LINIC (Olausson et al. 2023), CLOVER (Ryu et al. 2024), QuaSAR (Ranaldi, Valentino, and Freitas 2025), employ neuro-symbolic methods that integrate LLMs with symbolic reasoning to enhance the logic capacity of models. These methods abide by the following common workflow:

i) Employ the LLM to translate natural language into formal logic representations, including Deductive Reasoning, First-Order Logic, Constraint Satisfaction, *etc.*

ii) Entrust the reasoning mission to logic solvers, including Prover9 (McCune 2005), Pyke (Frederiksen 2008), Z3 (De Moura and Bjørner 2008), *etc.*

In these methods, the employed rigorous symbolic logic solvers ensure verifiable and faithful deductions (Step ii), but the suboptimal effect of these methods indicates the existence of weak links in their workflow, *i.e.* the translation from natural language to formal logic representations (Step i).

We empirically identify that the key issue in the translation (Step i) above is the absence of logical deduction, which is separately conducted by logic solvers in Step 2. Specifically, given a logic problem consisting of several premises

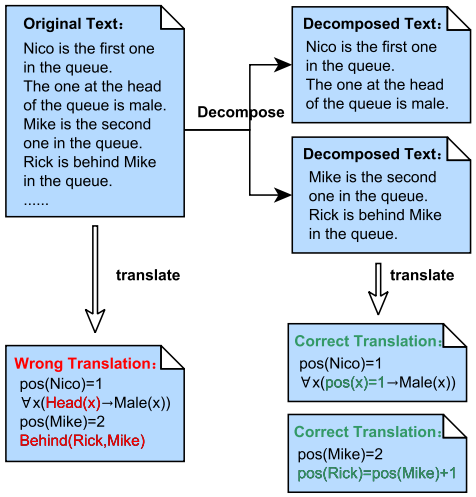


Figure 2: A demonstration of our empirical observations about logic translation. When given redundant premises carrying interference information, the model is unaware of the role of each premise in the deduction, thus reaching mindless logic translation that the logic solver cannot deduct upon; if we decompose the problem into understandable steps, the model catches the key point of this deduction, thus locating the crucial points in the problem statement and making effective translations for logic solvers.

and a question for an LLM to translate into formal representations, the model is unaware of the role of each premise in the deduction, **as it cannot see through the complex problem solution**. Please refer to Figure 2 for a demonstration. This prohibits the model from capturing crucial points in the problem statement, resulting in a suboptimal translation.

An intuitive solution is to make the problem more brief, *i.e.*, fewer premises that carry less interference information; thus, the translation model understands the key point of this deduction, and then it can locate the crucial points in the problem statement and make effective translations for logic solvers.

To demonstrate the effect of reducing the number of premises on better translation, we propose an empirical experiment where we employ LLMs to translate FOL problems into logic languages, then use the logic solver to solve the problem. The results are shown in Figure 3. As the logic solver is reliable and faithful, *i.e.*, a correct translation must lead to a correct answer, the accuracy reflects the performance of LLM translation. The results show that fewer premises are highly correlated with higher answer accuracy, which proves the effect of making the problem more brief for the translation model to see through.

On this basis, we propose the following method that recursively decomposes the problem into simple atomic deduction steps, which ensures an adequate problem understanding of the translation model.

Algorithm 1: Framework

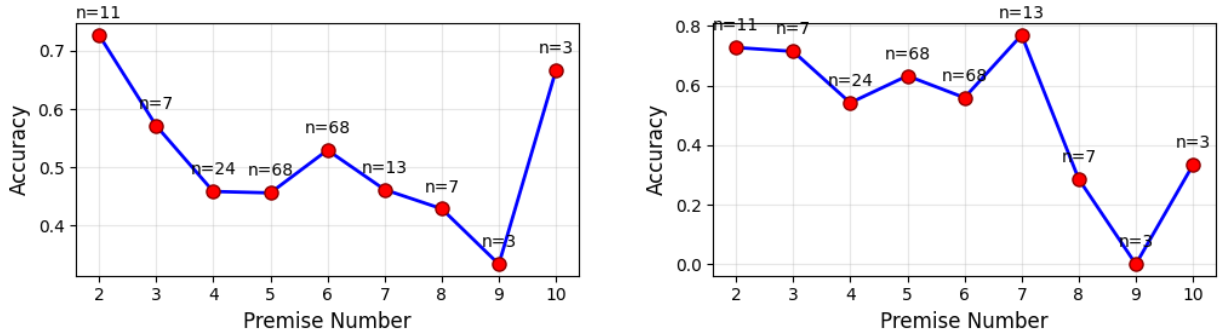
Input: Logic problem $P = (p_1, \ldots, p_k, q)$
Initialization: Deduction LLM $F(\cdot)$, logic extraction LLM $G(\cdot)$, translation LLM $H(\cdot)$, logic solver $L(\cdot)$, maximum disassemble depth d for Algorithm 2, maximum trial number T, trial experience $E = \emptyset$
Output: Answer to q

for t in range(T) **do**
  Disassembly: $\{S_1, \ldots, S_n\} = G(F(P))$, assume S_n is the final substep that reaches the answer to q.
  v_n = Verify(S_n) (Algorithm 2), this triggers a recursive verifiction consistent with the deduciton graph $G\{S_1, \ldots, S_n\}$, getting verifications v_i = Verify(S_i), $i = 1, \ldots, n-1$
  if $v_n = True$ **then**
    Break
  else
    Give feedback to the next trial: $E = E \bigcup \{(S_1, v_1), \ldots, (S_n, v_n)\}$
  end if
end for
return $S_n[conclusion]$

Algorithm 2: Verify

Input: A substep S_i to verify
Initialization: Logic problem $P = (p_1, \ldots, p_k, q)$, substeps $S_1, \ldots, S_n$, maximum disassemble depth d, translation LLM $H(\cdot)$, logic solver $L(\cdot)$.
Output: Verification of $S_i[conclusion]$, *i.e.* $v_n \in \{True, False, Uncertain\}$.

To verify S_i, gather required subconclusions $S = \{S_j | S_j[conclusion] \in S_i[proofs]\}$
for S_j in S **do**
  Verify subconclusion: v_j = Verify(S_j)
  if $v_j = Uncertain$ and d not exceeded **then**
    Further disassembly: $\{S'_1, \ldots, S'_k\} = G(F(S_j))$
    v_j = Verify(S'_k)
  end if
end for
Use only *True* subconclusions: $S_i[proofs] = S_i[proofs] \backslash \{S_j[conclusion] \in S_i[proofs] | v_j \neq True\}$
verification result $v_i = L(H(S_i))$
return v_i



GPT-3.5-turbo GPT-4

Figure 3: Accuracy of using different LLMs in FOL problems as translators.

Preliminaries

First-Order Logic (FOL) We employ the FOL grammar (Enderton 2001) in our method to parse natural languages into a verifiable logic language. Specifically, a logic problem (say P) is parsed into a list of FOL formulas including known *Premises* (say $p_1, \ldots, p_k$) and an unknown *Conclusion* (say q) to be proved.

LLM deduction Given an LLM and a logic problem $P = \{p_1, \ldots, p_k, q\}$ to solve, we formalize the natural language deduction of the LLM using substeps $\{S_1, \ldots, S_n\}$, $S_i = ([conclusion], [proofs])$, where $[conclusion]$ is the subconclusion reached in the substep, and $[proofs]$ denotes the premises and other subconclusions employed as its proofs. The subconclusion and its proofs in natural language can also be parsed into FOL formulas for verification. Note that this proof citation constructs a deduction graph $G(S_1, \ldots, S_n)$.

Method description

Our method decomposes the original problem into simpler deduction steps using LLMs, then verifies and solves the acquired substeps using the logic solver. If the substep carries unverifiable skipping of steps, we continue to decompose it for further verification recursively. The above disassembly and verification are conducted alternately until we reach a complete and verifiable deduction graph. Finally, we correct the deduction steps based on the verification results, and iterate the process above to reach an optimal deduction. Our framework is demonstrated in Figure 1 and summarized in Algorithm 1. We elaborate each part as follows.

Acquire deduction steps through recursive disassembly We disassemble the problem with the CoT(Chain-of-thought) (Sprague et al. 2024) technique to acquire substeps $\{S_1, \ldots, S_n\}$, $S_i = ([conclusion], [proofs])$. Formally, given a logic deduction problem $P = (p_1, \ldots, p_k, q)$, where $p_1, \ldots, p_k$ are premises, q is a question, we employ a deduction LLM $F(\cdot)$ with CoT to get a CoT deduction $F(P)$, and an extractor LLM $G(\cdot)$ with a prompt to extract substeps from the deduction, which gets substeps $\{S_1, \ldots, S_n\} = G(F(P))$.

The framework above leverages the deduction instinct of the LLM to simplify the problem, but it often carries unverifiable skipping of steps. If so, we continue to conduct the disassembly recursively, *i.e.* disassemble substep by doing $G(F(S_i))$, until we reach an atomic disassembly with verifiable substeps. This is described in Algorithm 2. Such verification ensures a faithful deduction, compensating for the faithfulness flaw of model deduction.

Translate and verify substeps To verify each substep, we first employ an LLM $H(\cdot)$ prompted to translate natural language into formal logic representations $H(S_i), i = 1, \ldots, n$, then we utilize the logic solver $L(\cdot)$ to give a verification result $v_i = L(H(S_i)) \in \{True, False, Uncertain\}$. Described in Algorithm 2 as a recursive function, the verification is complemented according to the deduction sequence in the deduction graph $G(S_1, \ldots, S_n)$. Note that **the verification of $S_i[conclusion]$ only uses *True* statements**, *i.e.* premises and verified *True* subconclusions in $S_i[proofs]$.

Feedback and adjustment To correct mistaken subconclusions or change the whole wrong thinking when it happens, we provide the model with the verification result that denotes the correct and incorrect subconclusions as feedback, which the model will take as a guiding experience for the next iteration. This is described in Algorithm 1.

Experiment

Experiment set-up

In this module, we introduced the basic settings used in our experimental method.

Datasets.

For logical reasoning problems, we evaluate our method and the baselines on the following datasets: FOLIO (Han et al. 2022), AR-LSAT (Zhong et al. 2022), Zebralogic (Lin et al. 2025), Puzzle (Srivastava et al. 2023), and PrOntoQA (Saparov and He 2022). The details of these datasets are introduced as follows.

FOLIO: The questions in this dataset are largely aligned with the frontiers of real-world knowledge and are phrased

in highly natural language. These questions are primarily solved through complex first-order logical reasoning. We selected all instances from this dataset to form the test set.

AR-LSAT: This dataset consists of analytical reasoning problems and is highly challenging, as even state-of-the-art baseline methods perform poorly on it. We selected all instances from this dataset for testing.

Zebralogic: This type of dataset belongs to multi-attribute sorting logic puzzles, which require logical reasoning based on all given clues to correctly assign each unique value of every attribute to its corresponding position (e.g., house). We randomly selected 200 instances from this dataset for testing.

Puzzle: This dataset covers a wide variety of reasoning problem types and is considered a comprehensive benchmark. We randomly filtered out 200 reasoning-related questions for testing.

PrOntoQA: This dataset is specifically designed to evaluate the deductive reasoning capabilities of large language models (LLMs). The task in PrOntoQA is to verify the truthfulness of a new fact. We used a instance from this dataset for testing.

Metrics. We employ accuracy (ACC) on the datasets as our primary evaluation metric. Additionally, the average time taken to solve a reasoning problem is computed and reported as a measure of method efficiency.

Compared Baselines. To demonstrate the advantages of our method, we select several recent and representative studies in related fields to serve as our baselines. These include Logic-LM (Pan et al. 2023), LINC (Olausson et al. 2023), CLOVER (Ryu et al. 2024), QuaSAR (Ranaldi, Valentino, and Freitas 2025), and SymCoT (Xu et al. 2024). Among them, SymCoT is a pure LLM-based method that leverages Chain-of-Thought (CoT), while the other approaches generally employ a framework that combines LLMs with logical solvers.

Implementation Details. In our experiments, all evaluations were conducted using LLMs and the Z3 solver following our framework's procedure. The maximum disassembly depth was set to 2 by default, and the maximum trail number was also set to 2 by default. These default values were determined by balancing the method's performance with computational overhead during runtime. Each step in the framework employs corresponding prompts, with detailed specifications provided in the Appendix .

Main Result.

Our main experimental results are presented in Figure 4. As demonstrated, the integration of LLMs with solvers yields significant improvements in reasoning capabilities compared to the standard LLM approach and the Chain-of-Thought (CoT) method. Specifically, the Chain-of-Thought (CoT) method improves the average accuracy by 6.1% compared to the standard approach. LogicLM demonstrates a 15.1% performance gain over the standard method, while our approach achieves a 32.7% improvement. Notably, our method maintains a 3.1% advantage over the state-of-the-art CLOVER approach.

Table 1: Performance of different methods applied to GPT 4 on logical reasoning datasets

Dataset	AR-LSAT	FOLIO	Zebralogic	Puzzle	PrOntoQA
Standard	0.329	0.662	0.125	0.620	0.765
CoT	0.342	0.701	0.140	0.605	0.915
Logic-LM	0.429	0.765	0.425	0.625	0.840
LINC	0.377	0.730	0.380	0.655	0.880
CLOVER	**0.641**	0.784	0.715	**0.825**	0.925
QuaSAR	0.558	0.745	0.405	0.755	0.855
SymCoT	0.442	0.775	0.155	0.640	0.890
Ours	0.632	**0.804**	**0.735**	0.820	**0.965**

Table 2: Performance of different methods applied to GPT3.5 on logical reasoning datasets

Dataset	AR-LSAT	FOLIO	Zebralogic	Puzzle	PrOntoQA
Standard	0.234	0.461	0.090	0.430	0.460
CoT	0.281	0.534	0.115	0.475	0.700
Logic-LM	0.381	0.642	0.345	0.520	0.725
LINC	0.320	0.627	0.315	0.515	0.680
CLOVER	0.541	0.706	**0.645**	0.740	0.775
QuaSAR	0.476	0.657	0.350	0.645	0.700
SymCoT	0.381	0.667	0.140	0.555	0.740
Ours	**0.571**	**0.745**	0.635	**0.800**	**0.875**

More importantly, while most baselines achieve accuracy rates of approximately 60-80% on simpler reasoning datasets such as FOLIO, Puzzle, and PrOntoQA, their performance drops significantly to around 40% on more challenging logical reasoning tasks like AR-LSAT and Zebralogic. This performance level is close to random guessing, indicating that these methods struggle with complex logical reasoning problems. Although CLOVER shows competitive performance on these difficult tasks, our method achieves the highest average accuracy of 63.1% and 69.8% on AR-LSAT and Zebralogic, respectively.

Furthermore, as shown in Tables 1 2 3, our method consistently achieves superior performance across different backbone models. Compared to the popular baseline Logic-LM, our approach improves average accuracy by 16.7%, 20.7%, and 15.6% on GPT-4, GPT-3.5, and DeepSeek models, respectively. When compared to the standard approach, our method demonstrates approximately $30\% \sim 40\%$ improvement across all models, highlighting its strong generalization capability across diverse model architectures.

Computational Cost

In this module, we evaluate the temporal cost of different methods. Our evaluation approach is as follows: we measure the average time required by each method to complete a logical reasoning problem using the GPT-4 model. Since the runtime of the SAT solver is negligible compared to that of the LLM, this metric is used to assess the computational cost.

The specific results are shown in Table 4. It can be observed that the average time for the Standard and CoT methods is 6–7 seconds, yet their accuracy is relatively low. The temporal costs of the other methods range from 10 sec-

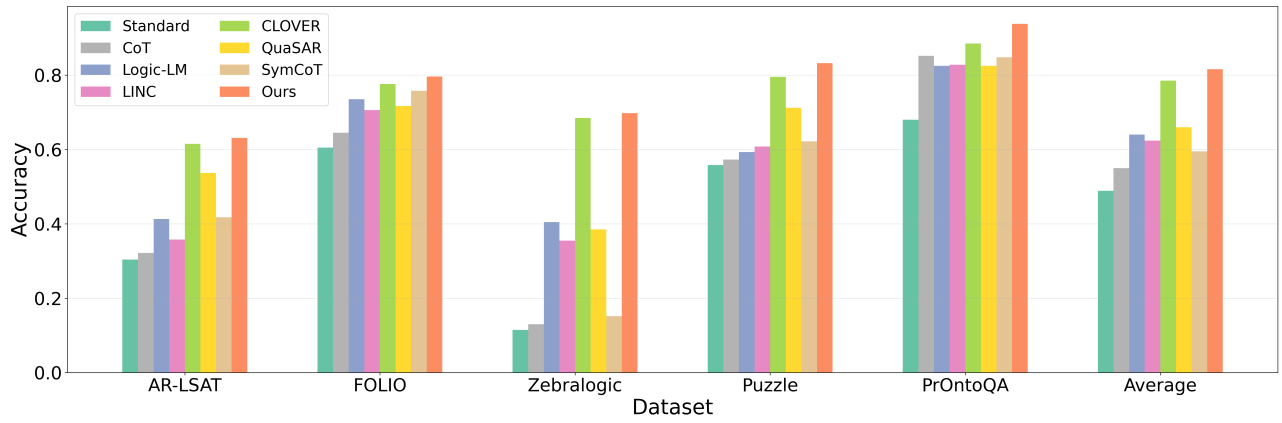


Figure 4: The average performance on three models of different methods

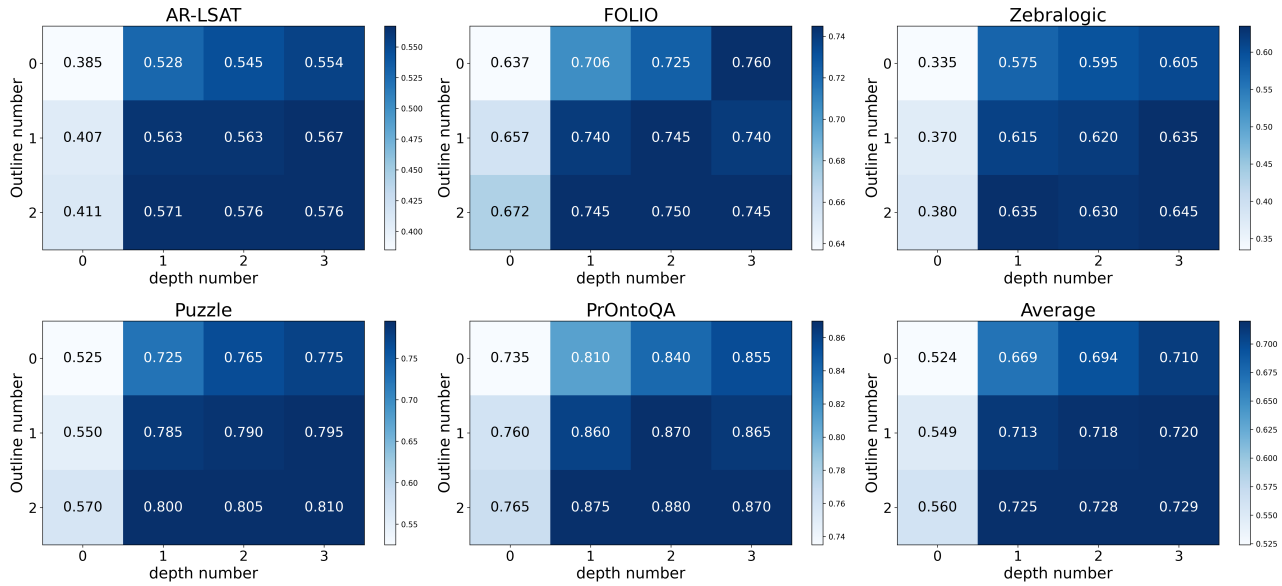


Figure 5: The performance of our method on GPT-3.5 using different combinations of hyperparameters

Table 3: Performance of different methods applied to Deepseek on logical reasoning datasets

Dataset	AR-LSAT	FOLIO	Zebralogic	Puzzle	PrOntoQA
Standard	0.351	0.691	0.130	0.625	0.815
CoT	0.342	0.701	0.135	0.640	0.940
Logic-LM	0.429	0.799	0.445	0.635	0.910
LINC	0.377	0.760	0.370	0.655	0.925
CLOVER	0.662	**0.838**	0.695	0.820	0.955
QuaSAR	0.576	0.750	0.400	0.735	0.920
SymCoT	0.433	0.833	0.160	0.670	0.915
Ours	**0.688**	0.838	**0.725**	**0.875**	**0.975**

onds to 100 seconds. Although our method incurs a higher temporal cost, its superior accuracy compensates for this drawback: compared to all baselines except CLOVER, our method achieves an average accuracy that is at least 15% higher, which fully justifies the increased temporal expenditure. Furthermore, while our method's average accuracy is only 3.1% higher than that of CLOVER, we simultaneously achieve a reduction of 23 seconds in temporal cost. Even

Table 4: Comparison of inference time costs using different methods with GPT-4.

Method	Standard	CoT	Logic-LM	LINC
Average time per question	6.3s	7.1s	26.8s	48.5s
Method	CLOVER	QuaSAR	SymCoT	Ours
Average time per question	99.2s	12.4s	51.7s	76.2s

when compared to the state-of-the-art method CLOVER, our approach maintains a dual advantage in both accuracy and temporal efficiency.

Ablation Study

In this module, we conduct ablation experiments to validate the importance and effectiveness of two key components in our method: decomposing the original problem into sub-problems and rethinking the outline. To this end, we define the maximum depth for problem decomposition as the hyperparameter *maximum disassemble depth*, and the maximum number of rethinking iterations as the hyper-

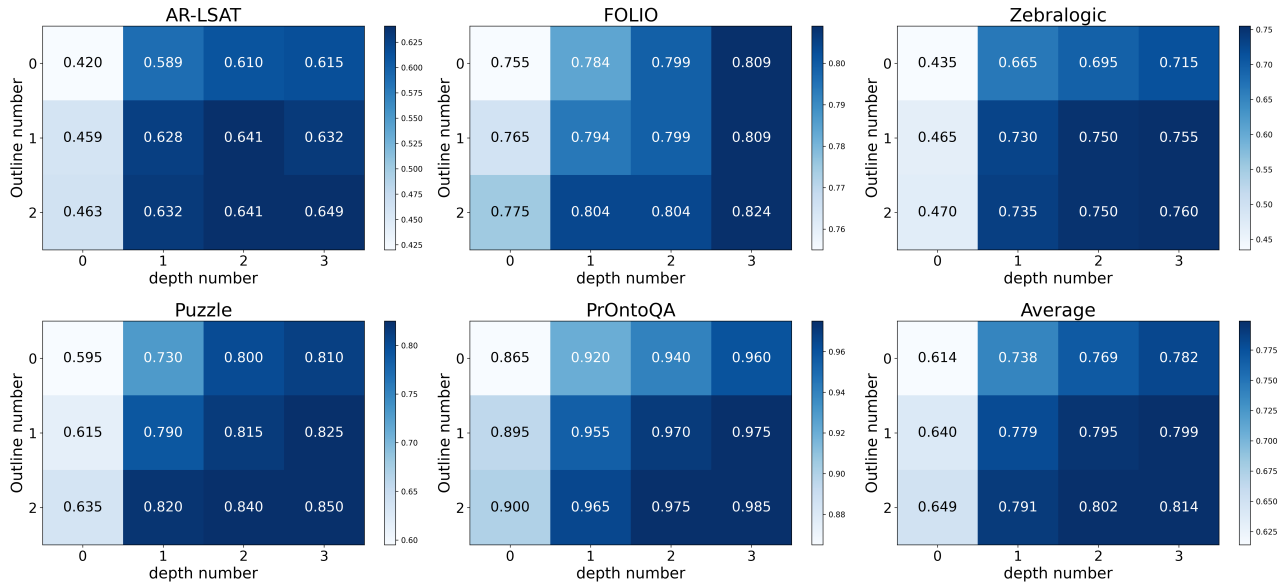


Figure 6: The performance of our method on GPT-4 using different combinations of hyperparameters

parameter *maximum trial number* . For convenience, we denote these two hyperparameters as *depth number* and *outline number*. In our experiments, the *depth number* takes values of 0, 1, 2, and 3, while the *outline number* takes values of 0, 1, and 2. We apply these hyperparameter configurations to both the GPT-4 and GPT-3.5 language models and conduct experiments on the default dataset. Results are illustrated in Figures 6 and Figures 5.

As shown in Figure 6, when *depth number* = 0, the language model degenerates into a logical problem translator. It can be observed that the average accuracy under this setting is suboptimal, with only 61.4% accuracy when *outline number* = 0. However, when the *outline number* increases to 1 and 2, the average accuracy improves by 2.6% and 3.5%, respectively. This indicates that during the process of outlining and rethinking, the model gains a deeper understanding of the problem, which subsequently enhances translation accuracy. Nevertheless, such improvements remain limited when *depth number* = 0. In contrast, when *depth number* increases to 1, even without outlining, the model exhibits a qualitative improvement in reasoning capability, with accuracy increasing by 12.4%. This validates our hypothesis that reasoning or translation errors often arise due to the excessive cognitive span of the original proposition. Decomposing complex propositions into combinations of simpler ones effectively enhances the model's reasoning performance. More importantly, when *depth number* = 1, increasing the *outline number* to 1 and 2 leads to accuracy improvements of 4.1% and 5.3%, respectively—significantly higher than the improvements observed when *depth number* = 0. This suggests a positive interaction between outlining and problem decomposition: outlining not only improves translation accuracy but also enhances the reasoning steps involved in subsequent problem decomposition. As shown in Figure 5, our experiments on GPT-3.5 further corroborate these findings, showing a 14.5% accuracy improvement when *depth number* increases to 1, along with a greater marginal benefit of increasing *outline number* compared to the *depth number* = 0 condition. Furthermore, when both depth and outline parameters are greater than or equal to 1, the accuracy shows a modest increase (within 3%) as the hyperparameters grow, sufficiently demonstrating the robustness of our method to variations in hyperparameter settings. Our method still exhibits parameter robustness on the Deepseek model; the experiment for Deepseek is detailed in Appendix B.

Conclusion

In this work, we critically examine the limitations of existing neuro-symbolic frameworks for logical reasoning, which often underutilize the capabilities of Large Language Models (LLMs) by relegating them to the role of mere text-to-symbol translators. This narrow usage fails to leverage the LLMs' inherent strengths in contextual understanding and problem structuring, while also leaving them susceptible to hallucinations and performance degradation in complex reasoning scenarios. To address these shortcomings, we proposed a novel decomposed collaborative framework that fosters a more profound and synergistic integration between LLMs and symbolic solvers. Our methodology strategically decomposes complex reasoning tasks into simpler, atomic sub-problems, allowing the LLM to dynamically analyze the problem, identify critical reasoning elements, and structure the logical relationships. The symbolic solver then rigorously verifies these atomic deductions, ensuring logical soundness and mitigating hallucinations. This iterative process, guided by feedback from the solver, enables the framework to progressively refine its reasoning path. In summary, our framework successfully harnesses the complementary strengths of LLMs and symbolic solvers, paving the way for more robust and trustworthy AI reasoning systems.

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Akbar, S. A.; Hossain, M. M.; Wood, T.; Chin, S.-C.; Salinas, E. M.; Alvarez, V.; and Cornejo, E. 2024. HalluMeasure: Fine-grained hallucination measurement using chain-of-thought reasoning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 15020–15037.

Barez, F.; Wu, T.-Y.; Arcuschin, I.; Lan, M.; Wang, V.; Siegel, N.; Collignon, N.; Neo, C.; Lee, I.; Paren, A.; et al. 2025. Chain-of-thought is not explainability. *Preprint, alphaXiv*, v1.

Chen, C.; and Shu, K. 2023. Can llm-generated misinformation be detected? *arXiv preprint arXiv:2309.13788*.

De Moura, L.; and Bjørner, N. 2008. Z3: An efficient SMT solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, 337–340. Springer.

Drori, I.; Zhang, S.; Shuttleworth, R.; Tang, L.; Lu, A.; Ke, E.; Liu, K.; Chen, L.; Tran, S.; Cheng, N.; et al. 2022. A neural network solves, explains, and generates university math problems by program synthesis and few-shot learning at human level. *Proceedings of the National Academy of Sciences*, 119(32): e2123433119.

Dunivin, Z. O. 2024. Scalable qualitative coding with llms: Chain-of-thought reasoning matches human performance in some hermeneutic tasks. *arXiv preprint arXiv:2401.15170*.

Enderton, H. 2001. *A Mathematical Introduction to Logic*. Academic Press. ISBN 9780122384523.

Frederiksen, B. 2008. Applying expert system technology to code reuse with pyke. *PyCon: Chicago*.

Golovneva, O.; Chen, M.; Poff, S.; Corredor, M.; Zettlemoyer, L.; Fazel-Zarandi, M.; and Celikyilmaz, A. 2022. Roscoe: A suite of metrics for scoring step-by-step reasoning. *arXiv preprint arXiv:2212.07919*.

Han, S.; Schoelkopf, H.; Zhao, Y.; Qi, Z.; Riddell, M.; Zhou, W.; Coady, J.; Peng, D.; Qiao, Y.; Benson, L.; et al. 2022. Folio: Natural language reasoning with first-order logic. *arXiv preprint arXiv:2209.00840*.

Kojima, T.; Gu, S. S.; Reid, M.; Matsuo, Y.; and Iwasawa, Y. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213.

Li, Y.; Du, Y.; Zhou, K.; Wang, J.; Zhao, W. X.; and Wen, J.-R. 2023. Evaluating object hallucination in large vision-language models. *arXiv preprint arXiv:2305.10355*.

Lin, B. Y.; Bras, R. L.; Richardson, K.; Sabharwal, A.; Poovendran, R.; Clark, P.; and Choi, Y. 2025. Zebralogic: On the scaling limits of llms for logical reasoning. *arXiv preprint arXiv:2502.01100*.

Liu, A.; Feng, B.; Xue, B.; Wang, B.; Wu, B.; Lu, C.; Zhao, C.; Deng, C.; Zhang, C.; Ruan, C.; et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Liu, H.; Ning, R.; Teng, Z.; Liu, J.; Zhou, Q.; and Zhang, Y. 2023. Evaluating the logical reasoning ability of chatgpt and gpt-4. *arXiv preprint arXiv:2304.03439*.

Luo, R.; Zheng, Z.; Wang, Y.; Ni, X.; Lin, Z.; Jiang, S.; Yu, Y.; Shi, C.; Chu, R.; Zeng, J.; et al. 2025. Ursa: Understanding and verifying chain-of-thought reasoning in multimodal mathematics. *arXiv preprint arXiv:2501.04686*.

Lyu, Q.; Havaldar, S.; Stein, A.; Zhang, L.; Rao, D.; Wong, E.; Apidianaki, M.; and Callison-Burch, C. 2023. Faithful chain-of-thought reasoning. In *The 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (IJCNLP-AACL 2023)*.

Manning, C. D. 2022. Human language understanding & reasoning. *Daedalus*, 151(2): 127–138.

McCarthy, J.; and Hayes, P. J. 1981. Some philosophical problems from the standpoint of artificial intelligence. In *Readings in artificial intelligence*, 431–450. Elsevier.

McCune, W. 2005. Release of prover9. In *Mile high conference on quasigroups, loops and nonassociative systems, Denver, Colorado*.

Mündler, N.; He, J.; Jenko, S.; and Vechev, M. 2023. Self-contradictory hallucinations of large language models: Evaluation, detection and mitigation. *arXiv preprint arXiv:2305.15852*.

Newell, A.; and Simon, H. 1956. The logic theory machine–A complex information processing system. *IRE Transactions on information theory*, 2(3): 61–79.

Olausson, T. X.; Gu, A.; Lipkin, B.; Zhang, C. E.; Solar-Lezama, A.; Tenenbaum, J. B.; and Levy, R. 2023. LINC: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. *arXiv preprint arXiv:2310.15164*.

Pan, L.; Albalak, A.; Wang, X.; and Wang, W. Y. 2023. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. *arXiv preprint arXiv:2305.12295*.

Ranaldi, L.; Valentino, M.; and Freitas, A. 2025. Improving chain-of-thought reasoning via quasi-symbolic abstractions. *arXiv preprint arXiv:2502.12616*.

Ribeiro, D.; Wang, S.; Ma, X.; Zhu, H.; Dong, R.; Kong, D.; Burger, J.; Ramos, A.; Wang, W.; Huang, Z.; et al. 2023. Street: A multi-task structured reasoning and explanation benchmark. *arXiv preprint arXiv:2302.06729*.

Ryu, H.; Kim, G.; Lee, H. S.; and Yang, E. 2024. Divide and translate: Compositional first-order logic translation and verification for complex logical reasoning. *arXiv preprint arXiv:2410.08047*.

Saparov, A.; and He, H. 2022. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *arXiv preprint arXiv:2210.01240*.

Sprague, Z.; Ye, X.; Bostrom, K.; Chaudhuri, S.; and Durrett, G. 2023. Musr: Testing the limits of chain-of-thought with multistep soft reasoning. *arXiv preprint arXiv:2310.16049*.

Sprague, Z.; Yin, F.; Rodriguez, J. D.; Jiang, D.; Wadhwa, M.; Singhal, P.; Zhao, X.; Ye, X.; Mahowald, K.; and Durrett, G. 2024. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning. *arXiv preprint arXiv:2409.12183*.

Srivastava, A.; Rastogi, A.; Rao, A.; Shoeb, A. A. M.; Abid, A.; Fisch, A.; Brown, A. R.; Santoro, A.; Gupta, A.; Garriga-Alonso, A.; et al. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on machine learning research*.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Wei, J.; Tay, Y.; Bommasani, R.; Raffel, C.; Zoph, B.; Borgeaud, S.; Yogatama, D.; Bosma, M.; Zhou, D.; Metzler, D.; et al. 2022a. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022b. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.

Xu, F.; Lin, Q.; Han, J.; Zhao, T.; Liu, J.; and Cambria, E. 2025. Are large language models really good logical reasoners? a comprehensive evaluation and beyond. *IEEE Transactions on Knowledge and Data Engineering*.

Xu, J.; Fei, H.; Pan, L.; Liu, Q.; Lee, M.-L.; and Hsu, W. 2024. Faithful logical reasoning via symbolic chain-of-thought. *arXiv preprint arXiv:2405.18357*.

Zhao, X.; Li, M.; Lu, W.; Weber, C.; Lee, J. H.; Chu, K.; and Wermter, S. 2023. Enhancing zero-shot chain-of-thought reasoning in large language models through logic. *arXiv preprint arXiv:2309.13339*.

Zhong, W.; Wang, S.; Tang, D.; Xu, Z.; Guo, D.; Chen, Y.; Wang, J.; Yin, J.; Zhou, M.; and Duan, N. 2022. Analytical reasoning of text. In *Findings of the Association for Computational Linguistics: NAACL 2022*, 2306–2319.

Appendix

A.Prompt Templates

1. Subgoals Prompt: This prompt instructs the model to deconstruct a given free-form deduction (<DEDUCTION>) into a sequence of structured steps. Each step must consist of a discrete subconclusion (<SUBCONCLUSION i>) and its corresponding proofs (<PROOFS>), which are references to premises or prior subconclusions. This structured format is crucial for enabling systematic, step-by-step verification of the reasoning chain.

Subgoals Prompt:

Solve the following First-Order Logic inference question:
<PREMISES>
<HINTS>
<QUESTION>
<DEDUCTION>
Please follow these guidelines:
- Summarize the deduction above by extracting the subconclusion and its proofs in each step.
- The line of the i-th subconclusion should start with "<SUBCONCLUSION i>" EXACTLY, and do NOT use line feed in one subconclusion.
- In the next line of each subconclusion, list its proofs using indices, like "premise i, premise j,..., subconclusion k, subconclusion l". This line should start with "<PROOFS>" EXACTLY, and do NOT use line feed.
- Each subconclusion is a CLEAN conclusion without its proof, analysis, and other problems.
- Denote the answer of true, false or uncertain with a line starting with "<ANSWER>", and also don't forget to list its proofs in the next line starting with <PROOFS>.
- Do NOT write subconclusions identical to any of the given premises and the question.
- Do NOT write subconclusions that are more complex than the original problems or contain other problems. The subconclusions should be SIMPLER than the given problem.
- Do NOT include proof, analysis, and other problems in each subconclusions.

<FEEDBACK>

2. Translation Prompt: This prompt is designed to convert a logic problem, originally expressed in natural language, into the precise syntactic format required by the Prover9 automated theorem prover. The model is provided with explicit formatting rules, including the placement of assumptions and goals, the correct use of logical operators (-, &, |—), and the structure of quantifiers (all, exists). This step bridges the gap between human-readable problem statements and machine-verifiable formalisms.

Translation Prompt:

Given the premises and a question, the task is to parse the premises and the question into a complete Prover9 input file.
The format of the Prover9 input file is as follows:
1) All premises must be placed between 'formulas(assumptions).' and 'end_of_list.'.
2) The conclusion to be proved must be placed between 'formulas(goals).' and 'end_of_list.'.
3) All formulas must end with a period (.).
4) Use '-' for negation, '&' for conjunction, and '—' for disjunction.
5) Use 'all x (...)' for universal quantification and 'exists x (...)' for existential quantification.
Premises:
All people who regularly drink coffee are dependent on caffeine.
People either regularly drink coffee or joke about being addicted to caffeine.
No one who jokes about being addicted to caffeine is unaware that caffeine is a drug.
Rina is either a student and unaware that caffeine is a drug, or neither a student nor unaware that caffeine is a drug.
If Rina is not a person dependent on caffeine and a student, then Rina is either a person dependent on caffeine and a student, or neither a person dependent on caffeine nor a student.
Question:
Based on the above information, is the following statement true, false, or uncertain? Rina is either a person who jokes about being addicted to caffeine or is unaware that caffeine is a drug.
formulas(assumptions).
all x (Drinks(x) → Dependent(x)).
all x (Drinks(x) — Jokes(x)).
all x (Jokes(x) → -Unaware(x)).
(Student(rina) & Unaware(rina)) — (-Student(rina) & -Unaware(rina)).
-(Dependent(rina) & Student(rina)) → ((Dependent(rina) & Student(rina)) — (-Dependent(rina) & -Student(rina))).
end_of_list.

formulas(goals).
Jokes(rina) — Unaware(rina).
((Jokes(rina) & Unaware(rina)) — (-Jokes(rina) & -Unaware(rina))) → (Jokes(rina) & Drinks(rina)).
end_of_list.
Premises:
Miroslav Venhoda was a Czech choral conductor who specialized in the performance of Renaissance and Baroque music.
Any choral conductor is a musician.
Some musicians love music.
Miroslav Venhoda published a book in 1946 called Method of Studying Gregorian Chant.
Question:
Based on the above information, is the following statement true, false, or uncertain? Miroslav Venhoda loved music.

formulas(assumptions).
Czech(miroslav) & ChoralConductor(miroslav) & Specialize(miroslav, renaissance) & Specialize(miroslav, baroque).
all x (ChoralConductor(x) → Musician(x)).
exists x (Musician(x) & Love(x, music)).
Book(methodOfStudyingGregorianChant) & Author(miroslav, methodOfStudyingGregorianChant) & Publish(methodOfStudyingGregorianChant, year1946).
end_of_list.

formulas(goals).
Love(miroslav, music).
exists y (exists x (Czech(x) & Author(x, y) & Book(y) & Publish(y, year1946))).
-exists x (ChoralConductor(x) & Specialize(x, renaissance)).
end_of_list.
——
Premises:
<PROBLEM>
Question:
<QUESTION>

3. Outline Prompt: This prompt directs the model to generate an initial, step-by-step deductive pathway to solve a given logical question. Based on the provided premises and hints, the model is tasked with producing a coherent line of reasoning. This output serves as the preliminary deduction that is subsequently analyzed and refined in later stages of our framework.

Outline Prompt:

Write a step-by-step deduction to solve the following question:
<PREMISES>
<HINTS>
<QUESTION>
<FEEDBACK>

4. Feedback Prompts: The following two prompts are employed when the model's initial attempt yields an incorrect result.

Answer Feedback Prompt:

TRIAL <TRIALID>
In this former trial, you gave the following summary, FAILURE. Please try again.
Your summary:
<SUMMARY>

Outline Feedback Prompt:

TRIAL <TRIALID>
In this former trial, you made the following subconclusions to reach the answer. Through rigid verification, we denote your correct (i.e., true) and incorrect (i.e., false or uncertain) subconclusions.
Your trial:
<DEDUCTION>
<SUBCONCLUSIONRESULTS>

B.More Experiments

In this section, we present hyperparameter experiments on the Deepseek model.

The empirical results are presented in Table 7. Through analysis, it can be found that the depth number is still an important parameter affecting the performance of the model. When both the outline number and depth number are 0, increasing the depth number by 1 increases the accuracy by 8.2%, and increasing the outline number by 1 increases the accuracy by 3.3%. When the outline number and depth number are greater than or equal to 1, the performance of the model ranges from 76% to 79%, proving that our method has a certain robustness to hyperparameters.

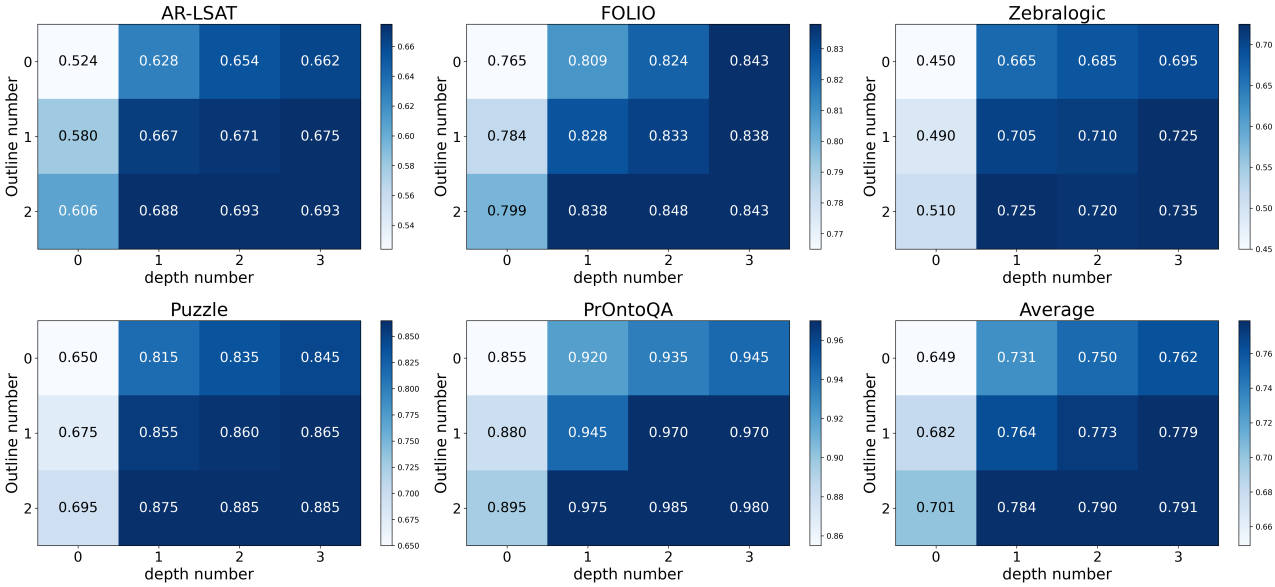


Figure 7: The performance of our method on Deepseeek using different combinations of hyperparameters