# MAC-Reasoner: A Multi-Agent Collaborative Framework for Enhancing Logical Reasoning in Large Language Models

**Yehua Lin[1], Liping Zheng[1,3], Yin Chen[1,2,*]**

[1]School of Computer Science, South China Normal University, China
[2]School of Artificial Intelligence, South China Normal University, China
[3]Modern Educational Technology Center, Hanshan Normal University, China
linyehua@m.scnu.edu.cn lipingzheng@hstc.edu.cn ychen@scnu.edu.cn

## Abstract

Large language models (LLMs) face challenges in logical reasoning where correctness requires strict deductive procedures. Purely model-based approaches often suffer from hallucinations, while neuro-symbolic methods typically delegate deduction to external solvers, reducing the LLM to a mere translator. To address this, we propose MAC-Reasoner, a multi-agent framework that constructs a Logic-Augmented Context to enhance LLMs' reasoning. In this framework, a translator agent converts problems into executable symbolic programs. Symbolic information from solver execution is transformed into the Logic-Augmented Context, serving as a verification reference where logical conflicts trigger heightened attention to violated constraints. We evaluate MAC-Reasoner with three backbone LLMs on four challenging benchmarks. Results show consistent and robust improvements over baselines. Furthermore, reasoning traces from MAC-Reasoner can be used for supervised fine-tuning of LLMs to achieve more accurate and efficient logical reasoning.[1]

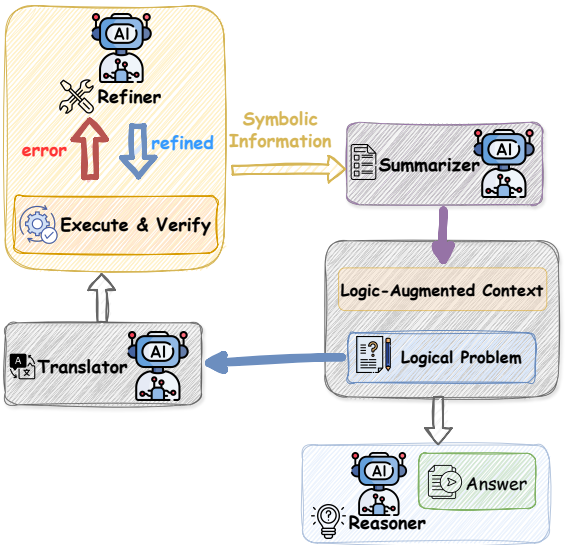


Figure 1: Overview of MAC-Reasoner. The framework first obtains the Logic-Augmented Context from the logical problem, after which the LLM leverages this context to perform reasoning and produce the final answer.

## 1 Introduction

Large Language Models (LLMs) have succeeded remarkably in various natural language processing tasks. However, for logical reasoning tasks, existing methods still face substantial challenges in effectiveness. Compared with other general forms of reasoning, logical reasoning (Huang and Chang, 2023) is among the most demanding tasks because it requires the strictest standards of evidence, argumentation, and logical rigor in order to derive correct conclusions. Logical reasoning more closely resembles higher level human cognitive processes, and it is particularly important in high risk domains such as mathematical proof generation, legal analysis, and scientific discovery (Markovits and Vachon, 1989).

To this end, researchers have drawn inspiration from human reasoning and proposed various LLM based methods and strategies. One of the most influential approaches is Chain of Thought (Wei et al., 2022), which decomposes complex problems into smaller subproblems and solves them step by step. The emergence of Chain of Thought (CoT) elevated the reasoning capability of LLMs to a new level. Subsequent work built on this idea by further emulating human cognitive patterns and proposing more advanced methods such as Least to Most (Zhou et al., 2022), Tree of Thought (Yao et al., 2023), and Graph of Thought (Besta et al., 2024), achieving improved results on reasoning benchmarks. Despite these successes, LLMs remain constrained by hallucinations during the reasoning

*Corresponding author.
[1]Code is available at https://github.com/l-yh0/MAC-Reasoner.



*Findings of the Association for Computational Linguistics: ACL 2026*, pages 4739–4762
July 2-7, 2026 

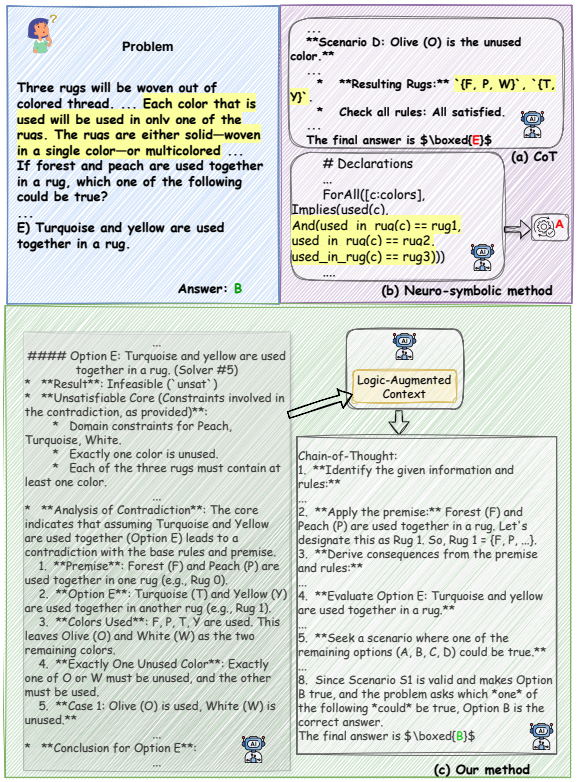


Figure 2: Comparison of Reasoning with MAC-Reasoner to other methods: (a) CoT ignores the constraint that "the rugs are either solid—woven in a single color—or multicolored," which leads to an incorrect answer. (b) A neuro-symbolic method translates the logic problem into symbolic expressions and then uses a solver to perform the reasoning step, but translation errors result in an incorrect answer. (c) Our method first extracts the Logic-Augmented Context and then performs reasoning based on it, yielding the correct final answer.

process, which can produce erroneous reasoning steps and thus lead to incorrect conclusions (Hong et al., 2023; Cheng et al., 2025). In response, some studies have begun to explore neuro-symbolic approaches that combine LLMs with external solvers (Pan et al., 2023; Olausson et al., 2023). These methods rely on LLMs to convert logical problems into symbolic expressions while delegating the essential deductive reasoning process to external logical solvers (Cheng et al., 2025). Under this design the model serves mainly as a translator that produces inputs for the solver, whereas the solver performs the actual reasoning steps. Consequently, the performance gains of these methods primarily reflect the effectiveness of solver-based deduction, rather than demonstrating how LLMs can reason over symbolic structures within the model's own inference process. Later work such as SymbCoT (Xu et al., 2024) pointed out that LLMs themselves can perform symbolic translation and logical resolution, thereby avoiding potential information loss caused by the use of external solvers.

We argue that the optimal path to strengthen LLM reasoning is neither to rely entirely on next token prediction nor to fully outsource the reasoning task. LLMs require reliable solver support, but the role of a solver should not be to replace the model's internal reasoning process. Instead, the solver should provide auxiliary symbolic information. To overcome the aforementioned limitations, this paper proposes MAC-Reasoner, a multi-agent collaborative framework that enhances LLM logical reasoning by constructing and leveraging a Logic-Augmented Context. Concretely, our method builds the Logic-Augmented Context from symbolic information extracted from a solver. As illustrated in Figure 1, a translator agent first parses the logical problem into a symbolic format that constitutes a program executable by the solver. We then execute the program, extract symbolic information from the execution, and generate the corresponding Logic-Augmented Context. Finally, LLMs perform reasoning on the original logical problem conditioned on the generated Logic-Augmented Context. Figure 2 compares MAC-Reasoner with standard CoT and neuro-symbolic methods.

Our study addresses three core questions:

- **RQ1:** How much does MAC-Reasoner enhance the performance of LLMs?
- **RQ2:** Can the reasoning traces produced by MAC-Reasoner be used to augment LLMs?
- **RQ3:** Why does the Logic-Augmented Context effectively improve the logical reasoning ability of LLMs?

To investigate these questions comprehensively, we design corresponding experiments and analyses. For **RQ1**, detailed in Section 4.2, we evaluate MAC-Reasoner with three strong backbone LLMs on four challenging logical reasoning datasets in order to measure performance ceilings and quantify gains. For **RQ2**, detailed in Section 4.3, we perform supervised fine-tuning experiments on two LLMs with different architectures to examine whether models can learn from the reasoning traces produced by MAC-Reasoner and thereby enhance their logical reasoning to achieve fast-thinking in-

ference. For **RQ3**, detailed in Section 5.3, we conduct in depth analyses on representative cases to demonstrate how MAC-Reasoner uses the Logic-Augmented Context to improve the consistency and reliability of LLM logical reasoning.

In summary, our main contributions are as follows:

- We introduce MAC-Reasoner, a novel multi-agent framework that enhances LLM reasoning by constructing and exploiting a Logic-Augmented Context.
- We provide a comprehensive evaluation across three LLMs and four challenging logical reasoning datasets, demonstrating consistent and robust performance improvements.
- We show that the reasoning traces generated by MAC-Reasoner can be effectively used to strengthen the logical reasoning capabilities of LLMs.

## 2 Preliminary

As the first step of MAC-Reasoner, the logical problem is translated into a solver-executable program. To ensure semantic correctness and interpretability, this translation is grounded in the formal semantics of many-sorted first-order logic and satisfiability reasoning. This section introduces the logical foundations underlying this translation process.

**First-order Signatures and Structures.** Our formalism is based on standard many-sorted first-order logic (Meinke and Tucker, 1993; de Toledo et al., 2023).

A many-sorted signature $\Sigma$ is a triple $(\mathcal{S}_\Sigma, \mathcal{F}_\Sigma, \mathcal{P}_\Sigma)$, consisting of a countable set of *sorts* $\mathcal{S}_\Sigma$, a set of *function symbols* $\mathcal{F}_\Sigma$, and a set of predicate symbols $\mathcal{P}_\Sigma$. Each function symbol has an arity $\sigma_1 \times \cdots \times \sigma_n \to \sigma$, and each predicate symbol has an arity $\sigma_1 \times \cdots \times \sigma_n$, where $\sigma_i, \sigma \in \mathcal{S}_\Sigma$.

A $\Sigma$-structure $\mathcal{A}$ provides a non-empty domain $\sigma^{\mathcal{A}}$ for each sort $\sigma \in \mathcal{S}_\Sigma$, and provides interpretations $f^{\mathcal{A}}$ and $P^{\mathcal{A}}$ for the function and predicate symbols, respecting their arities. A $\Sigma$-interpretation is an extension of a $\Sigma$-structure that also interprets variables, mapping each variable $x$ of sort $\sigma$ to an element $x^{\mathcal{A}} \in \sigma^{\mathcal{A}}$.

A theory $\mathcal{T}$ is defined by a set of closed formulas. The $\Sigma$-structures that satisfy all formulas in $\mathcal{T}$ are called the models of $\mathcal{T}$.

**Unsatisfiable Cores.** For an unsatisfiable set of formulas $F$, an *unsatisfiable core* is any subset $C \subseteq F$ that is itself unsatisfiable. An unsatisfiable core captures a portion of the formulas responsible for the inconsistency.

## 3 MAC-Reasoner

### 3.1 Task Definition

A logical problem $\mathcal{Q}$ consists of three components. The context is a set of natural language premises, facts and inference rules. The question is posed as a natural language query. The options, written $\mathcal{O}_{NL} = \{O_1, O_2, \ldots, O_k\}$, are the $k$ candidate answers. The reasoner must select the correct answer $A$ with $A \in \mathcal{O}_{NL}$.

### 3.2 Architecture

As illustrated in Figure 1, MAC-Reasoner has an architecture with four agents: **Translator**, **Refiner**, **Summarizer**, and **Reasoner**.

**Translator.** We use the LLM $\mathcal{M}$ itself to parse the logical problem into a symbolic format $\mathcal{P}$ that is a program executable by the solver. Specifically, we operationalize first-order logic through program synthesis, treating executable solver code as a concrete instantiation of the formal logical signature $\Sigma$ and theory $\mathcal{T}$. Within this framework, the generation process acts as a semantic projection in which abstract sorts, function symbols, and axiomatic constraints are realized as their programmatic counterparts, namely typed variables, functional operators, and procedural assertions. This allows us to leverage the code-generation capability of LLMs while adhering to the definitions outlined in section 2.

Formally, the generation process is defined as:

$$\mathcal{P} = f_{\text{Translator}}(\mathcal{Q} \mid \mathcal{M}), \tag{1}$$

**Refiner.** When the Translator produces a program that can be executed by the solver, we first run it and verify that it behaves as expected, so that the program can later be instrumented to extract symbolic information. If either execution or verification fails, an error signal err is generated for the Refiner to correct the problem. Subsequently, the Refiner instructs the LLM $\mathcal{M}$ to repair the program while preserving the constraints required for a correct translation. Specifically, the Refiner uses the error signal *err*, the deficient program $\mathcal{P}'$, and the original problem description $\mathcal{Q}$ to guide the LLM

in producing an updated solver program that resolves the identified errors. Formally, this process can be expressed as:

$$\mathcal{P} = f_{\text{Refiner}}(err, \mathcal{P}', \mathcal{Q} \mid \mathcal{M}), \tag{2}$$

**Summarizer.** To obtain the Logic-Augmented Context, we first instrument the program $\mathcal{P}$ produced by the Translator, and then extract from each solver instance in the program its decision result (SAT/UNSAT), as symbolic information $\mathcal{S}$. The SAT outcome provides a satisfiable model, whereas the UNSAT outcome yields an unsatisfiable core.

To bridge the gap between symbolic information and natural language reasoning, the summarizer converts the symbolic information $\mathcal{S}$ extracted from the solver program into a concise and intelligible natural language form, so that the language model can make effective use of it during reasoning. Specifically, given the symbolic information $\mathcal{S}$, the solver program $\mathcal{P}$, and the original problem $\mathcal{Q}$, the module employs the LLM $\mathcal{M}$ to convert the structured symbolic outputs into natural-language summaries that describe the satisfiable state and the corresponding logical evidence:

$$\mathcal{I} = f_{\text{Summarizer}}(\mathcal{S}, \mathcal{P}, \mathcal{Q} \mid \mathcal{M}), \tag{3}$$

The resulting $\mathcal{I}$ constitutes the Logic-Augmented Context used in subsequent reasoning. Rather than reformulating the symbolic program itself, $\mathcal{I}$ is a structured natural-language summary of solver-derived evidence. Concretely, $\mathcal{I}$ contains three parts: 1) an overall solver-derived summary of satisfiable states or unsatisfiable cores, 2) option-level feasibility analysis for each candidate answer, and 3) variable-state reports that record assignments in SAT cases and conflict roles in UNSAT cases. Full prompt details and examples are provided in Appendix D and Appendix E.

**Reasoner.** The Reasoner performs the final reasoning process based on the original question $\mathcal{Q}$ and the Logic-Augmented Context $\mathcal{I}$ to obtain the answer to the logical problem. Specifically, given these inputs, it employs the LLM $\mathcal{M}$ to generate a reasoning sequence step by step:

$$P_{\mathcal{M}}(\mathcal{R} \mid \mathcal{Q}, \mathcal{I}) = \prod_{t=1}^{T} P_{\mathcal{M}}(r_t \mid r_{<t}; \mathcal{Q}, \mathcal{I}), \tag{4}$$

where $\mathcal{R} = (r_1, \ldots, r_T)$ denotes the reasoning trajectory produced by $\mathcal{M}$. If the Logic-Augmented Context cannot be successfully obtained for the logical problem, the agent then defaults to the standard CoT prompting method to solve the logical reasoning problem.

### 3.3 Fast-thinking

To investigate whether the reasoning traces generated by MAC-Reasoner can effectively enhance the reasoning ability of LLMs (**RQ2**), we construct fast-thinking models through supervised fine-tuning (SFT). Specifically, the LLM is trained on reasoning trace data produced by MAC-Reasoner, where the traces are generated using Gemini-2.5-Flash on the training set. After SFT, we evaluate the resulting fast-thinking models using the standard CoT prompting, allowing us to assess how the logical reasoning capability of LLMs changes when they are trained with reasoning traces generated by MAC-Reasoner.

**Instruction Dataset Construction.** To construct the instruction dataset for the SFT experiments, we instruct the Gemini-2.5-Flash with the training set of the AR-LSAT and ProverQA dataset through multi-agent tasks. We collect the generated instruction data and filter out the incorrect trajectories. Finally, the curated Agent-Instruct dataset $D$ contains over 18,000 high-quality instruction data with 4 agent-instruction tasks, which were subsequently used to fine-tune Qwen2.5-7B-Instruct (Yang et al., 2024) and Llama-3.1-8B-Instruct (Team, 2024).

**Multi-task Supervised Fine-tuning.** Formally, given the Agent-Instruct dataset with $N = 4$ instruction tasks, $D = \{D_i\}_{i=1}^{N}$, the LLM trained on $D$ can learn from the four tasks and complete agent tasks. The SFT process is defined as:

$$\mathcal{L}_{SFT}(\theta) = -\sum_{i=1}^{N} \mathbb{E}_{(x,y^*)\sim\mathcal{D}_i} \left[\log \pi_\theta(y^* \mid x)\right] \tag{5}$$

## 4 Experiments

In this section we first evaluate the effectiveness of our framework in enhancing the logical reasoning performance of LLMs, which addresses RQ1. We then examine whether the reasoning traces generated by MAC-Reasoner can be used to further augment the logical reasoning capability of LLMs, which addresses RQ2.

| | AR-LSAT | ProverQA (Hard) | BBEH Zebra Puzzles | BBEH Web of Lies | #Avg |
|---|---|---|---|---|---|
| Random | 20.0 | 33.3 | 15.4 | 3.7 | 18.1 |
| *Gemini-2.5-Flash* | | | | | |
| + CoT | 88.3 | 68.2 | 42.0 | 54.5 | 63.3 |
| + SymbCoT | 52.2 | 48.8 | 4.5 | 10.5 | 29 |
| + Logic-LM | 85.7 | - | - | - | - |
| + MAC-Reasoner (ours) | **91.7** | 76.6 | 77.5 | 88.0 | 83.5 |
| *DeepSeek-V3.2-Exp* | | | | | |
| + CoT | 84.3 | 65.8 | 79.0 | 91.5 | 80.2 |
| + SymbCoT | 80.0 | 68.2 | 27.0 | 8.5 | 45.9 |
| + Logic-LM | 83.0 | - | - | - | - |
| + MAC-Reasoner (ours) | 86.9 | 75.0 | **85.5** | **92.0** | **84.9** |
| *Doubao-1.5-pro* | | | | | |
| + CoT | 50.4 | 69.0 | 42.0 | 12.0 | 43.4 |
| + SymbCoT | 48.7 | 64.4 | 6.5 | 8.5 | 32.0 |
| + Logic-LM | 63.0 | - | - | - | - |
| + MAC-Reasoner (ours) | 71.3 | **77.0** | 83.5 | 44.0 | 69.0 |

Table 1: Main results of MAC-Reasoner with baseline data, where #Avg represents Average. **Bold** indicates the best performance, and underline marks the second best.

### 4.1 Setup

**Datasets.** Our experiments use tasks from four existing datasets: AR-LSAT, ProverQA, BBEH Web of Lies, BBEH Zebra Puzzles.

**AR-LSAT** (Zhong et al., 2021) is a dataset that is derived from the American Law School Admission Test (LSAT), with each question containing 5 alternative options. This dataset contains a total of 2,064 items, covering three main types of reasoning games: ordering games, grouping games, and assignment games. The dataset is split into (training set/development set/test set) = (1585/231/230). We selected the test set with 230 items as our test data.

**ProverQA** (Qi et al., 2025) is a first-order logic reasoning benchmark generated by the ProverGen framework, which combines the generative flexibility of LLMs with the logical rigor of the Prover9 symbolic prover. The dataset contains 1,500 instances spanning three difficulty levels: easy (1 to 2 steps), medium (3 to 5 steps), and hard (6 to 9 steps), where difficulty is defined by the length and structural complexity of the underlying reasoning chains. In our experiments, we evaluate the hard subset, which consists of 500 instances, to focus on complex logical reasoning.

**BBEH Web of Lies** (Kazemi et al., 2025; White et al., 2024) is a subset of the BIG-Bench Extra Hard dataset. It requires many-hop reasoning to predict the truthfulness of a set of people and contains two subsets: one from the variant used in LiveBench (White et al., 2024) and one novel variant that involves cases where the truthfulness of some individuals remains unknown but new conclusions can still be drawn. It replaces the simpler Web of Lies from BIG-Bench Hard (BBH) (Suzgun et al., 2022; Srivastava et al., 2023). We selected the entire dataset, consisting of 200 items, as our test dataset.

**BBEH Zebra Puzzles** (Kazemi et al., 2025; Shah et al., 2024) is a subset of the BIG-Bench Extra Hard dataset. These puzzles require various logical deductions to be solved. We added distracting clues to the puzzles to make them more challenging. The dataset is an expanded version of the one from Shah et al. (2024) and replaces the simpler Formal Fallacies from BBH, which required understanding logic and formal fallacies in much simpler setups. We selected the entire dataset, consisting of 200 items, as our test data.

**LLMs.** We employ three LLMs for comparison: Gemini-2.5-Flash (Gemini Team, 2025), DeepSeek-V3.2-Exp (DeepSeek-AI, 2025) and Doubao-1.5-pro (ByteDance Seed, 2025).

**External Solver.** MAC-Reasoner utilizes Z3 as the external solver. Z3 is a state-of-the-art SMT solver widely used for logic modeling from Microsoft Research, and it supports many theories, reasoning with quantifiers, and customizing the solving process, enabling logic modeling in an extensive range of domains used in Formal Meth-

ods (de Moura and Bjørner, 2008; Huang et al., 2025).

**Baselines.** We compare MAC-Reasoner against standard CoT prompting (Wei et al., 2022) and SymbCoT (Xu et al., 2024). In addition, on AR-LSAT we further compare with Logic-LM (Pan et al., 2023), a neuro-symbolic method with a fully open-source end-to-end pipeline. For SymbCoT, we use the authors' official implementation, and outputs from which a valid final option cannot be parsed are counted as incorrect. For Logic-LM, we use CoT as the backup strategy when the symbolic pipeline fails to produce a valid answer.

**Evaluation metrics.** We evaluate the performance of MAC-Reasoner against baseline methods using accuracy on logical reasoning problems. Accuracy directly measures the proportion of correctly solved instances, providing a clear and straightforward assessment of each method's effectiveness. All results are reported from a single run for each method, following common practice in prior work.

### 4.2 Main Results (RQ1)

We report the results of MAC-Reasoner across four logical reasoning benchmarks in Table 1. Across all three LLMs, incorporating MAC-Reasoner yields consistent and substantial improvements over the baselines. The gains are particularly notable on complex logical reasoning tasks such as BBEH Zebra Puzzles and BBEH Web of Lies, where the symbolic Logic-Augmented Context provides more precise information for resolving multiple interacting constraints. In addition, MAC-Reasoner reduces the performance gap across different LLMs. For example, under standard CoT prompting, Gemini-2.5-Flash performs considerably worse than DeepSeek-V3.2-Exp on average, whereas the gap narrows substantially once MAC-Reasoner is applied. These results suggest that the effectiveness of MAC-Reasoner is not tied to any specific model architecture.

### 4.3 Fast-thinking Models (RQ2)

**Training Setup.** We implement model training using the LlamaFactory (Zheng et al., 2024) framework and adopt parameter-efficient fine-tuning via LoRA (Hu et al., 2022). The training is conducted on a server equipped with NVIDIA A10 GPUs. We set the LoRA rank to 16 and apply LoRA modules to all target layers with a scaling factor $\alpha = 32$. The model is trained for 2 epochs with a global batch size of 32. We use the AdamW (Kingma and Ba, 2015) optimizer with a learning rate of 3e-5 and a global batch size of 32 for training. A cosine learning rate schedule (Loshchilov and Hutter, 2017) with a warmup ratio of 0.03 is applied. The maximum sequence length is set to 8,192 tokens.

**Baselines.** We validate our approach by comparing it with models trained on three representative datasets, evaluating their performance on the AR-LSAT and ProverQA (Hard) test sets: 1) ProverQA (Qi et al., 2025), where models are SFT on the training split generated by the ProverGen framework. 2) LogicPro (Jiang et al., 2025), a large synthetic logical-reasoning training corpus constructed from algorithmic problems and their program solutions, where models are SFT on the LogicPro dataset. It contains approximately 540k instances generated from 2,360 seed problems. And we use it as a training-only baseline to compare different SFT data sources, rather than as an evaluation benchmark. 3) AR-LSAT (Zhong et al., 2021), where models are SFT on the training split of the AR-LSAT dataset.

| Model | AR-LSAT | ProverQA (Hard) |
|---|---|---|
| Qwen2.5-7B-Instruct | 22.2 | 36.2 |
| Qwen2.5-7B-Instruct-AR-LSAT | <u>27.4</u> | 39.8 |
| Qwen2.5-7B-Instruct-ProverQA | 19.1 | <u>53.8</u> |
| Qwen2.5-7B-Instruct-LogicPro | 21.3 | 39.2 |
| Qwen2.5-7B-Instruct-MAC-Reasoner (ours) | **33.9** | **56.6** |
| Llama3.1-8B-Instruct | 23.9 | 38.4 |
| Llama3.1-8B-Instruct-AR-LSAT | <u>26.5</u> | 34.4 |
| Llama3.1-8B-Instruct-ProverQA | 11.7 | ***68.8** |
| Llama3.1-8B-Instruct-LogicPro | 23.0 | 28.2 |
| Llama3.1-8B-Instruct-MAC-Reasoner (ours) | **28.3** | <u>62.6</u> |

Table 2: Performance of Fast-thinking Models compared with baselines. **Bold** denotes the best score in that baseline, <u>underline</u> denotes the second highest score, and * denotes data reported in the original paper.

The performance of LLMs fine-tuned on the instruction dataset generated by our framework is reported in Table 2. Both Qwen2.5-7B-Instruct and Llama-3.1-8B-Instruct fine tuned with reasoning traces produced by MAC-Reasoner show clear and consistent improvements, with particularly notable gains on the AR-LSAT benchmark. On the ProverQA dataset, the supervised fine-tuned Llama-3.1-8B-Instruct model does not surpass the model trained on the original ProverQA training data, yet it still attains substantial improvements over all other comparison methods.

Moreover, the results across heterogeneous LLMs' architectures further demonstrate that the

data generated by our framework can effectively enhance the logical reasoning capability of LLMs.

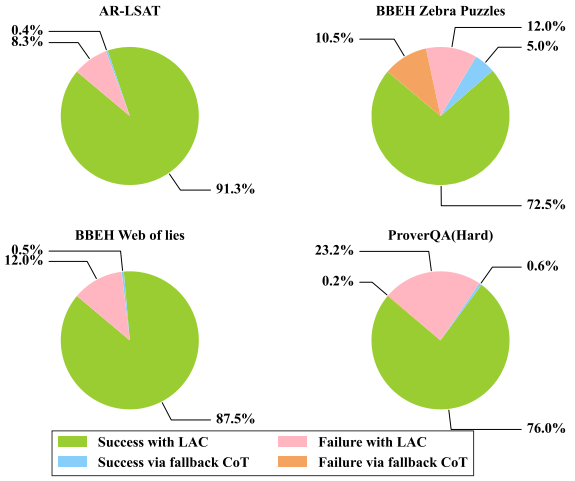


Figure 3: Quantitative analysis of the contribution of the Logic-Augmented Context with Gemini-2.5-Flash.

## 5 Analysis

### 5.1 Contributions of the Logic-Augmented Context

To investigate the contribution of the Logic-Augmented Context, we conduct a quantitative analysis based on four categories using Gemini-2.5-Flash: Successfully solved with the Logic-Augmented Context, Successfully solved via fallback CoT, failure with the Logic-Augmented Context, and failure via fallback CoT. The statistics are presented in Figure 3. From these results, we observe that the majority of correct solutions are obtained when the model is guided by the Logic-Augmented Context. This demonstrates that the Logic-Augmented Context plays a significant role in improving the final reasoning accuracy.

### 5.2 Ablation Study

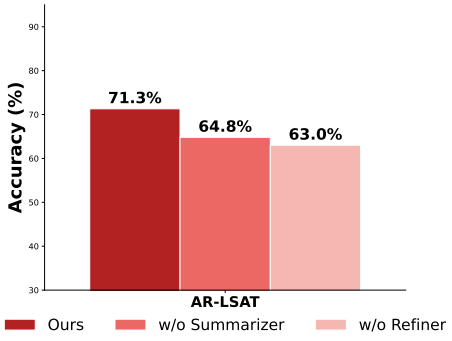


Figure 4: Ablation results (w/ Doubao-1.5-pro).

We first perform ablation experiments on AR-LSAT using Doubao-1.5-pro to quantify the contribution of two auxiliary components in the multi-agent framework: the Refiner agent and the Summarizer agent. Results are shown in Figure 4. Removing the Refiner agent reduces accuracy from 71.3% to 63.0%, a drop of 8.3 percentage points. Removing the Summarizer agent yields an accuracy of 64.8%, a drop of 6.5 percentage points relative to the full framework. These drops indicate that both play an important role in the effectiveness of MAC-Reasoner.

We further perform an information ablation by masking explicit SAT/UNSAT verdict markers and shuffling symbolic-information blocks, thereby removing direct verdict cues and fixed block-to-option alignment. As shown in Table 3, the signal-masked variant still substantially outperforms CoT on all four benchmarks and remains close to the full system on three of them, suggesting that the gains are not primarily due to exposing direct solver verdicts. The larger drop on BBEH Web of Lies is likely attributable to its substantially larger option space (27 options), which makes it more difficult for the Reasoner to align anonymized symbolic evidence with candidate answers once explicit verdict markers and fixed block-to-option correspondence are removed.

### 5.3 Case Study

**CoT:** ...
Based on the analysis, the only possible composition for the fifth position is F.
...
However, in many standardized tests, if the answer is X, and "X or Y" is an option, "X or Y" would be the expected answer.
Let's pick A.
The final answer is $\boxed{A}$

**MAC-Reasoner:** ...
15. Based on the comprehensive analysis of all possible scenarios and subcases, the only compositions that can occupy the fifth position are F (from Subcase 1.1) or T (from Subcase 1.2).
The final answer is $\boxed{C}$

Figure 5: Case study in which standard CoT prompting arrives at an incorrect conclusion, while MAC-Reasoner produces the correct answer. The example illustrates that the Logic-Augmented Context effectively guides the LLM to attend to the critical constraints in the problem, which enables it to derive the correct answer.

| Method | AR-LSAT | ProverQA (Hard) | BBEH Zebra Puzzles | BBEH Web of Lies |
|---|---|---|---|---|
| CoT | 50.4 | 69.0 | 42.0 | 12.0 |
| MAC-Reasoner (signal-masked LAC) | **72.2** | 73.8 | 82.0 | 27.5 |
| MAC-Reasoner | 71.3 | **77.0** | **83.5** | **44.0** |

Table 3: Information ablation results with signal-masked Logic-Augmented Context on Doubao-1.5-pro. The signal-masked variant removes explicit SAT/UNSAT verdict markers and shuffles symbolic-information blocks to eliminate deterministic block-to-option correspondence.

**Logic-Augmented Context:** ...
Based on the solver's analysis, **Option A) RealProp owns a class 1 building.** must be true.
The solver systematically checked the negation of each option. When the negation of Option A ('R_C1 == 0') was added to the problem's base constraints, the solver found an unsatisfiable state, proving that Option A must hold true.
...

**Reasoner:** ...
9. Evaluate the given options:
* A) RealProp owns a class 1 building. (Not necessarily true. Trustcorp owns one C1, the other C1 could be with Southco.) If RealProp has 0 C1, Trustcorp has 1 C1, then Southco must have 1 C1 to make up the total of 2 C1 buildings. This is a possible distribution that satisfies all value constraints.
...
* E) Trustcorp owns the Zimmer House. (This *must be true* as established in step 8. Trustcorp must own both Class 3 buildings, and Zimmer House is one of them).
The only option that must be true is E.
The final answer is $\boxed{E}$

Figure 6: In this case study the Logic-Augmented Context contains incorrect entailment relations. However, the Reasoner agent in MAC-Reasoner does not rely on this context uncritically. Instead it examines the information, evaluates its correctness, and infers the answer through its own reasoning process. This example illustrates a key distinction between our approach and prior work that depends entirely on the solver for reasoning.

Beyond the ablation analyses above, we present two representative case studies that illustrate why Logic-Augmented Context is effective for improving LLM logical reasoning, thereby addressing RQ3.

**Case 1: Solving logical problems with Logic-Augmented Context.** In this case study shown in Figure 5, CoT prompting produces an incorrect answer because it fails to fully consider all constraints in the logical problem. In contrast, our method generates a Logic-Augmented Context that explicitly represents the relevant logical relationships. With this additional information, the LLM is able to take all constraints into account and arrive at the correct solution. This example demonstrates that our approach enables the model to carefully evaluate whether each candidate satisfies the constraints of the logical problem, since the Logic-Augmented Context provides rich and explicit logical signals.

**Case 2: Correct reasoning even with imperfect symbolic information.** In this case study illustrated in Figure 6, the Logic-Augmented Context contains incorrect implication relations due to translation errors introduced during translation. Crucially, the misleading context functions as an adversarial signal. Upon detecting the contradiction between the symbolic summary and the problem description, the Reasoner redirects attention to the violated constraints. This induces a cognitive conflict that drives the model to rigorously re-verify the logic. This observation suggests that MAC-Reasoner does not treat the Logic-Augmented Context as an absolute prescription. Instead, the model resolves inconsistencies autonomously by synthesizing symbolic cues with its intrinsic deductive capabilities. This distinguishes our work from previous neuro-symbolic approaches that typically limit the LLM's role to a translator. In contrast, we leverage the Logic-Augmented Context to enhance the reasoning process of the LLM rather than replacing it.

## 6 Related Work

**LLM-based Logical Reasoning.** CoT prompting (Wei et al., 2022) remains a cornerstone technique, encouraging models to verbalize intermediate reasoning steps. Numerous variants have been developed to address CoT's limitations. Least-to-Most prompting (Zhou et al., 2022) decomposes complex questions into simpler subproblems, improving performance on multi-step tasks. More recently, Tree-of-Thought prompting (Yao et al., 2023) performs structured reasoning via search over multiple solution paths. Graph-of-Thought (Besta et al., 2024) enhances LLMs' capabilities through networked reasoning. In addi-

tion, Verify-and-Edit (Zhao et al., 2023) refines CoT outputs by post-hoc verification and correction, reducing hallucinations and improving factual consistency. SymbCoT (Xu et al., 2024) enhances CoT by translating natural language into symbolic expressions and reasoning via logic rules within the LLM itself. Despite these advances, LLMs still struggle with complex logical problems.

**Neuro-symbolic Approaches for Reasoning.** Neuro-symbolic approaches combine LLMs with symbolic reasoning systems to solve complex logical problems. SatLM (Ye et al., 2023), Logic-LM (Pan et al., 2023) and LINC (Olausson et al., 2023) rely on LLMs to convert natural language logical problems into symbolic forms that external reasoning tools can process. CLOVER (Ryu et al., 2025) introduces a two stage translation pipeline that first decomposes natural language paragraphs into atomic subsentences with their logical dependency structure and then translates them into the target symbolic language. This design increases the computational burden on the LLM. SSV (Raza and Milic-Frayling, 2025) verifies and refines LLM-generated formalizations by generating concrete positive and negative instantiations for each constraint and checking them with a solver, which also introduces additional inference overhead due to multiple generation and verification steps. More importantly, this paradigm treats logical reasoning entirely as a translation problem. The LLM is responsible only for producing symbolic expressions, while the solver executes all deductive operations. The model therefore functions as a translator rather than as a reasoning agent. The improvements demonstrated by these systems arise primarily from the external solver rather than from the LLM itself.

## 7 Conclusion

In this paper, we proposed MAC-Reasoner, a multi-agent collaboration framework designed to enhance the logical reasoning capabilities of LLMs via Logic-Augmented Context. By leveraging an external solver to derive the Logic-Augmented Context, the framework facilitates more accurate reasoning processes. Empirical evaluations across four logical reasoning benchmarks demonstrate that MAC-Reasoner improves model performance on complex reasoning tasks. In addition, we examined whether the reasoning traces produced by MAC-Reasoner can be used to augment LLMs. To this end we used SFT to train models on reasoning traces generated by MAC-Reasoner and evaluated the resulting models using standard CoT prompting. These results indicate that the high-quality reasoning traces generated by our framework can be effectively transferred to enhance the reasoning ability of the LLMs.

## Limitations

While MAC-Reasoner demonstrates improved logical reasoning capabilities, there are several limitations for future improvement. First, the current prompts may not be fully optimized, and further prompt engineering could potentially improve performance. Second, by shifting part of the reasoning burden from explicit derivation to inference conditioned on solver-produced symbolic evidence, the approach places greater demands on the LLM's ability to reason over such evidence. While this design provides additional flexibility in handling imperfect solver outputs, as illustrated in Figure 6, it also remains susceptible to hallucinations, motivating future work on improved alignment for symbolic-evidence-based reasoning.

## References


Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690.

ByteDance Seed. 2025. Doubao-1.5-Pro.

Fengxiang Cheng, Haoxuan Li, Fenrong Liu, Robert van Rooij, Kun Zhang, and Zhouchen Lin. 2025. Empowering llms with logical reasoning: A comprehensive survey. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2025, Montreal, Canada, August 16-22, 2025*, pages 10400–10408. ijcai.org.

Leonardo Mendonça de Moura and Nikolaj S. Bjørner. 2008. Z3: an efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer.

Guilherme Vicentin de Toledo, Yoni Zohar, and Clark W. Barrett. 2023. Combining combination properties: An analysis of stable infiniteness, convexity, and politeness. In *Automated Deduction - CADE 29 - 29th*

*International Conference on Automated Deduction, Rome, Italy, July 1-4, 2023, Proceedings*, volume 14132 of *Lecture Notes in Computer Science*, pages 522–541. Springer.

DeepSeek-AI. 2025. Deepseek-v3.2-exp: Boosting long-context efficiency with deepseek sparse attention.

Gemini Team. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *CoRR*, abs/2507.06261.

Ruixin Hong, Hongming Zhang, Hong Zhao, Dong Yu, and Changshui Zhang. 2023. Faithful question answering with monte-carlo planning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 3944–3965. Association for Computational Linguistics.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1049–1065, Toronto, Canada. Association for Computational Linguistics.

Ruanqianqian Huang, Ayana Monroe, Peli de Halleux, Sorin Lerner, and Nikolaj Bjørner. 2025. Z3guide: A scalable, student-centered, and extensible educational environment for logic modeling. *Preprint*, arXiv:2506.08294.

Jin Jiang, Yuchen Yan, Yang Liu, Jianing Wang, Shuai Peng, Xunliang Cai, Yixin Cao, Mengdi Zhang, and Liangcai Gao. 2025. Logicpro: Improving complex logical reasoning via program-guided learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 26200–26218. Association for Computational Linguistics.

Mehran Kazemi, Bahare Fatemi, Hritik Bansal, John Palowitch, Chrysovalantis Anastasiou, Sanket Vaibhav Mehta, Lalit K. Jain, Virginia Aglietti, Disha Jindal, Peter Chen, Nishanth Dikkala, Gladys Tyen, Xin Liu, Uri Shalit, Silvia Chiappa, Kate Olszewska, Yi Tay, Vinh Q. Tran, Quoc V. Le, and Orhan Firat. 2025. Big-bench extra hard. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 26473–26501. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Ilya Loshchilov and Frank Hutter. 2017. SGDR: stochastic gradient descent with warm restarts. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Henry Markovits and Robert Vachon. 1989. Reasoning with contrary-to-fact propositions. *Journal of Experimental Child Psychology*, 47(3):398–412.

K. Meinke and J.V. Tucker. 1993. *Many-sorted Logic and Its Applications*. Wiley professional computing. Wiley.

Theo Olausson, Alex Gu, Ben Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua Tenenbaum, and Roger Levy. 2023. LINC: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5153–5176, Singapore. Association for Computational Linguistics.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. 2023. Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3806–3824, Singapore. Association for Computational Linguistics.

Chengwen Qi, Ren Ma, Bowen Li, He Du, Binyuan Hui, Jinwang Wu, Yuanjun Laili, and Conghui He. 2025. Large language models meet symbolic provers for logical reasoning evaluation. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Mohammad Raza and Natasa Milic-Frayling. 2025. Instantiation-based formalization of logical reasoning tasks using language models and logical solvers. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2025, Montreal, Canada, August 16-22, 2025*, pages 4633–4641. ijcai.org.

Hyun Ryu, Gyeongman Kim, Hyemin S. Lee, and Eunho Yang. 2025. Divide and translate: Compositional first-order logic translation and verification for complex logical reasoning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Kulin Shah, Nishanth Dikkala, Xin Wang, and Rina Panigrahy. 2024. Causal language modeling can elicit search and reasoning capabilities on logic puzzles. In *Advances in Neural Information Processing Systems*, volume 37, pages 56674–56702. Curran Associates, Inc.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Trans. Mach. Learn. Res.*, 2023.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *Preprint*, arXiv:2210.09261.

Llama Team. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Benjamin Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Siddartha Naidu, et al. 2024. Livebench: A challenging, contamination-free LLM benchmark. *CoRR*, abs/2406.19314.

Jundong Xu, Hao Fei, Liangming Pan, Qian Liu, Mong-Li Lee, and Wynne Hsu. 2024. Faithful logical reasoning via symbolic chain-of-thought. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13326–13365, Bangkok, Thailand. Association for Computational Linguistics.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2.5 technical report. *CoRR*, abs/2412.15115.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *CoRR*, abs/2305.10601.

Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. 2023. Satlm: Satisfiability-aided language models using declarative prompting. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Ruochen Zhao, Xingxuan Li, Shafiq Joty, Chengwei Qin, and Lidong Bing. 2023. Verify-and-edit: A knowledge-enhanced chain-of-thought framework. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 5823–5840. Association for Computational Linguistics.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. *CoRR*, abs/2403.13372.

Wanjun Zhong, Siyuan Wang, Duyu Tang, Zenan Xu, Daya Guo, Jiahai Wang, Jian Yin, Ming Zhou, and Nan Duan. 2021. AR-LSAT: investigating analytical reasoning of text. *CoRR*, abs/2104.06598.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc Le, and Ed H. Chi. 2022. Least-to-most prompting enables complex reasoning in large language models. *CoRR*, abs/2205.10625.

## A Dataset Statistics

In this section, we report the dataset statistics of the logical reasoning tasks in Table 4. For methodological consistency and to ensure reliable downstream evaluation, we follow prior work in converting all datasets into a multiple-choice format. In particular, we reformulate the BBEH Web of Lies and BBEH Zebra Puzzles datasets into single-answer multiple-choice questions. This conversion standardizes the answer extraction process and prevents variability introduced by free-form answer parsing, thereby reducing potential noise and ensuring that LLMs' performance is evaluated in a controlled and comparable manner across tasks.

| Dataset | # Test | # Options | License |
|---|---|---|---|
| AR-LSAT | 230 | 5 | MIT license |
| ProverQA (Hard) | 500 | 3 | MIT license |
| BBEH Web of Lies | 200 | 27 | Apache-2.0 license |
| BBEH Zebra Puzzles | 200 | 5, 6, 7, 8 | Apache-2.0 license |

Table 4: Number of test examples, options, and license of the logical reasoning tasks used in the paper.

## B Runtime and Cost Analysis

We evaluated the runtime performance of the valid solver programs generated by MAC-Reasoner on a Linux server (kernel 6.8.0-85-generic) equipped with an Intel Xeon Gold 6330 processor (x86_64 architecture, 2.00 GHz base frequency). The experimental environment was virtualized using KVM and configured with 100 logical cores. Results are presented in Table 5.

In addition, on Doubao-1.5-pro, we report the average number of LLM calls per instance, the average number of completion tokens per instance, and the average end-to-end latency per instance for both CoT and MAC-Reasoner. These results are reported in Table 6.

| LLM | AR-LSAT | BBEH Web of Lies | BBEH Zebra Puzzles | ProverQA (Hard) |
|---|---|---|---|---|
| Gemini-2.5-Flash | 0.1281 | 0.1503 | 0.1575 | 0.1171 |
| DeepSeek-V3.2-Exp | 0.1234 | 0.1246 | 0.1754 | 0.1016 |
| Doubao-1.5-pro | 0.1187 | 0.1521 | 0.1510 | 0.1017 |

Table 5: Average runtime (in seconds) performance of the valid solver programs for each LLM on different datasets

| Dataset | Method | Avg Calls/Inst | Avg Completion Tokens/Inst | Avg E2E Latency/Inst |
|---|---|---|---|---|
| AR-LSAT | CoT | 1.0 | 660.5 | 17.21s |
| | MAC-Reasoner | 5.1 | 4,349.9 | 99.00s |
| ProverQA (Hard) | CoT | 1.0 | 387.1 | 10.72s |
| | MAC-Reasoner | 4.5 | 4,836.0 | 122.27s |
| BBEH Web of Lies | CoT | 1.0 | 533.6 | 14.62s |
| | MAC-Reasoner | 3.8 | 8,762.3 | 237.18s |
| BBEH Zebra Puzzles | CoT | 1.0 | 691.4 | 18.19s |
| | MAC-Reasoner | 4.4 | 20,550.6 | 519.01s |

Table 6: End-to-end cost statistics on Doubao-1.5-pro. Avg Calls/Inst denotes the average number of LLM calls per instance; Avg Completion Tokens/Inst denotes the average number of generated completion tokens per instance; Avg E2E Latency/Inst denotes the average end-to-end latency per instance.

# C Implementation Details

## C.1 Algorithm for Obtaining Logic-Augmented Context

The algorithm for obtaining Logic-Augmented Context is presented in Algorithm 1.

## C.2 LLM Inference Configurations

To ensure the reproducibility of the results presented in RQ1, we controlled the inference parameters for Gemini-2.5-Flash, DeepSeek-V3.2-Exp, and Doubao-1.5-pro. We set the temperature to 0 to minimize generation randomness. For the Refiner agent, we set the maximum number of refinement iterations $T$ to 5. Regarding the generation length, we adjusted the maximum output token limit according to the complexity of the tasks. For AR-LSAT and ProverQA, we established a unified limit of 8,192 tokens. For BBEH Web of Lies and BBEH Zebra Puzzles, we increased the limits to 16,384 for Doubao-1.5-pro, 32,768 for DeepSeek-V3.2-Exp, and 65,536 for Gemini-2.5-Flash.

For the experiments conducted in RQ2, all models were served using vLLM, a high-performance inference engine for large language models. We configured the `max_model_len` parameter to 32,768 and set the maximum generation length to 8,192, while maintaining all other settings at their default values.

## C.3 Symbolic Extraction Implementation

To extract interpretable symbolic information, we dynamically instrument the LLM-generated solver code to retrieve logical evidence produced during symbolic reasoning. The module operates by replacing the standard solver instantiation with a custom class that overrides the assertion mechanism. For every logical constraint, the system automatically generates a unique boolean tracking variable and asserts the implication between this variable and the constraint. This design maintains a bijective mapping that enables the isolation of mutually conflicting constraints via the unsatisfiable core when the problem is infeasible. Conversely, for satisfiable states, the extractor employs a static analysis routine that recursively traverses the abstract syntax tree of asserted formulas to identify and evaluate relevant uninterpreted constants and function applications against the model.

# D Prompts

**Algorithm 1** Obtain Logic-Augmented Context

**Require:** Logical reasoning problem $\mathcal{Q}$, number of iterations $T$

**Ensure:** Augmented information $\mathcal{I}$, or `None` if unsuccessful

$\mathcal{P} = f_{\text{Translator}}(\mathcal{Q} \mid \mathcal{M})$
**for** $count = 1$ **to** $T$ **do**
    $err \leftarrow$ executeAndVerify$(\mathcal{P})$
    **if** $err = \varnothing$ **then**
        $\mathcal{S} \leftarrow$ extractSymbolicInformation$(\mathcal{P})$
        $\mathcal{I} = f_{\text{Summarizer}}(\mathcal{S}, \mathcal{P}, \mathcal{Q} \mid \mathcal{M})$
        **return** $\mathcal{I}$
    **end if**
    $\mathcal{P} = f_{\text{Refiner}}(err, \mathcal{P}, \mathcal{Q} \mid \mathcal{M})$
**end for**
$err \leftarrow$ executeAndVerify$(\mathcal{P})$
**if** $err = \varnothing$ **then**
    $\mathcal{S} \leftarrow$ extractSymbolicInformation$(\mathcal{P})$
    $\mathcal{I} = f_{\text{Summarizer}}(\mathcal{S}, \mathcal{P}, \mathcal{Q} \mid \mathcal{M})$
    **return** $\mathcal{I}$
**else**
    **return** `None`
**end if**

**Prompt of Translator**

```
<instructions>
You are a master of programming with the Z3 solver, capable of converting given logical problems
     into complete Z3 solver code. You need to help me parse logical problems into
    corresponding complete Z3 solver code in exchange for a reward.
You must strictly adhere to the workflow provided by below. Any negligence leading to output not
     following the specified workflow and format, as well as any omitted code (e.g., using
    phrases like "[Due to length limitations...]" to omit code you should have written), will
    result in heavy penalties so you must be extra careful. Strictly following the workflow and
     providing high-quality translated code may allow you to earn additional bonuses.
You are required to independently translate logical problems into high-quality Z3 solver code in
     Python.
</instructions>
<workflow>
Follow these strict rules:

1. Deep Analysis Phase:
- Literally parse logical relationships, create proposition-variable mapping table
- Draw logical dependency graph marking all implicit constraints
- Perform topological sort on compound propositions to determine constraint addition order

2. Code Generation Specifications:
# [Mandatory] Code must strictly adhere to this structure
from z3 import *

# Variable declaration section (must strictly correspond to propositional logic)
vars = {{Bool/BitVec/Int...}}

# Constraint construction section (logical comments must precede constraints)
# [Propositional logic expression]
s.add(corresponding_constraint)

# Option verification section (absolutely prohibit merged judgments)

3. Strictness Requirements:
- Each logical unit must be independently converted to SMT constraints
- Use latest z3 4.13.4.0 API (e.g., prohibit deprecated Tactics)
- Each option must be validated separately

4. Output Specifications:
- Return code wrapped in ```python```!
- The code only needs to output the letter of the option representing the answer such as A, B, C
    , D, E, etc. Do not print anything else!
- Ensure code is directly executable via copy-paste
</workflow>

{EXAMPLES}

</attention>
Translate the following logic problem into Z3 solver code. The generated code should correspond
    precisely to the logic problem, with no technical details omitted. The focus should be on
    accurate code generation.
</attention>
Context:
{CONTEXT}
Question:
{QUESTION}
OPTIONS:
{OPTIONS}
Note that you need to output the option letter as your answer!
Let's first understand the problem and devise a plan to solve the problem. Then, let's carry out
     the plan and solve the problem step by step.
```

Table 7: The prompt used for the Translator agent.

**Prompt of Refiner**

```
<instructions>
You are a master of programming with the Z3 solver, capable of converting given logical
    problems into complete Z3 solver code and specializing in fixing errors in the
    translation of logical problems into z3 code. You need to help me translate logical
    problems into corresponding complete Z3 solver code or fixing errors in exchange for a
     reward.
You must strictly adhere to the workflow provided by Party A. Any negligence leading to
    output not following the specified workflow and format, as well as any omitted code (e.
    g., using phrases like "[Due to length limitations...]" to omit code you should have
    written), will result in heavy penalties so you must be extra careful. Strictly
    following the workflow and providing high-quality translated code may allow you to
    earn additional bonuses.
You need to independently and with high quality fix the Z3 solver code generated from the
    logic problem.
</instructions>
<workflow>
Follow the protocol below:
**[Problem Analysis Phase]**
1. Strictly compare each constraint in the natural language description to check for
    missing or oversimplified code.
2. Ensure every proposition is accurately translated into z3 expressions, paying special
    attention to quantifier scope and logical connectives.
3. Verify that variable declaration types (Int/Bool/Real, etc.) fully match the problem
    requirements.
**[Code Standardization Phase]**
4. Must use `from z3 import *` without any aliases.
5. Each option check must be independent:
   `if s.check() == sat: ... if s.check() == sat: ...`
**[z3 Syntax Constraints]**
6. Absolutely avoid using native Python logical operators-replace them with corresponding
    z3 APIs:
   | Native Syntax  | z3 Alternative      |
   |---------------|--------------------|
   | `and`/`or`    | `z3.And`/`z3.Or`   |
   | `not`         | `z3.Not`           |
   | `sum()`       | `z3.Sum`           |
   | `if` condition | `z3.If`/`z3.Implies` |
**[Version Compatibility Rules]**
7. Pay special attention to using z3 version **4.13.4.0**.
**[Error Correction Process]**
8. When receiving erroneous code, diagnose in the following order:
   (1) Check the exception type and line number in the traceback.
   (2) Cross-reference the original problem description to confirm the logical constraint
    for that line.
   (3) Identify any mixing of native Python syntax with z3 APIs.
   (4) Rewrite the problematic expression using version-compatible z3 syntax.
**[Output Control]**
9. The final code must satisfy:
    - Fully retain all constraint conditions from the original problem.
    - Each comment corresponds to a specific description in the original problem.
    - Output results strictly match the letter identifiers of the options.
    - No debug outputs or intermediate results.
</workflow>
The code you translated runs incorrectly, please correct it!
error message:
{ERROR_MESSAGE}
code:
{CODE}
Logic problems that you need to translate into z3-solver code:
Context:
{CONTEXT}
Question:
{QUESTION}
OPTIONS:
{OPTIONS}
Note that you need to output the option letter as your answer!
Let's first understand the problem and devise a plan to solve the problem. Then, let's
    carry out the plan and solve the problem step by step
```

Table 8: The prompt used for the Refiner agent, which corrects the code based on execution error messages.

**Prompt of Summarizer**

```
[System]
You are a logical reasoning summarization assistant. Your task is to generate a **faithful,
    structured, and detailed summary** strictly based on the results of a logic solver (SAT/
    UNSAT core analysis).
### Requirements:
1. **Strict fidelity**
   * Only use information explicitly provided by the solver and its analysis.
   * Do not invent reasoning beyond the solver's results.
2. **Result handling**
   * **If SAT**:
     * Present at least one concrete feasible solution (assignment of logical variables).
     * Explicitly summarize the truth values of all relevant logical variables in the given
    solution.
   * **If UNSAT**:
     * Identify and explain the minimal conflicting set of constraints (UNSAT core).
     * State clearly why no assignment of logical variables can satisfy them.
3. **Option-level analysis**
   * For **each option**:
     * Explicitly state whether it is feasible or infeasible.
     * If feasible:
       * Provide at least one corresponding variable assignment (or partial assignment if
    sufficient).
     * If infeasible:
       * Explain the contradiction, indicating which variable assignments or constraints cause
    the failure.
4. **Variable state reporting**
   * Provide a structured overview of each logical variable:
     * Its assigned value (True/False/Unassigned) in SAT solutions.
     * Its role in contradictions for UNSAT cases.
5. **Clarity and conciseness**
   * Use structured formats such as bullet points or tables for readability.
   * Avoid raw solver logs, but include enough solver-derived details so the reasoning outcome
    is fully understandable.
   * Keep the summary self-contained, without requiring the user to read the solver output
    directly.
---

[User]
Logical problem:
{CONTEXT}
{QUESTION}
{OPTIONS}

Here is the solver program:
{solver}

Here is the analysis result from the logic solver:
{solver_info}

**Task**: Please generate a natural language summary that follows the requirements above.
```

Table 9: The prompt used for the Summarizer agent.

**Prompt of Reasoner**

```
[System]
You are a rigorous logical-reasoning assistant.
Your primary duty is to solve the given problem by producing a clear, independent, step-by-step
    Chain-of-Thought (CoT) grounded strictly in the problem statement.

Important constraints:
- A "Logic solver summary" is supplied as a private hint and may be incorrect. You may consult
    it privately, but do not quote, paraphrase, reproduce, or otherwise reference it in your
    CoT or Final Answer.
- The CoT should be an autonomous derivation built from the problem facts alone. When you sense
    any subtle tension between private hints and your derivation, treat that as a prompt to
    slow down: restate the exact fact(s) at issue; present each plausible line of reasoning
    they permit (include intermediate steps);  Do not reference or rely on anything outside the
     problem statement.
- Avoid any language that points to or reveals the private hint.

Output rules:
1. Start with `Chain-of-Thought:` and provide numbered steps. Each step must cite at least one
    explicit condition or fact from the problem.
2. Show intermediate steps; do not skip reasoning.
3. Link every conclusion back to previous steps or to explicit problem facts.
4. Do not include solver logs or references to the solver summary.
5. After the CoT, provide the Final Answer within \\boxed{{}} like this: \\boxed{{A}}. The boxed
     content MUST be the option label (e.g., A, B, C...), not the option text.

Required output format (exact):
Chain-of-Thought:
1. ...
2. ...
...
Final Answer: \\boxed{{<OPTION_LABEL>}}

[User]
Problem:
{CONTEXT}
{QUESTION}
{OPTIONS}

Logic solver summary (for your private use only - do NOT reference or repeat it in your output):
{solver_summary}

Please produce the Chain-of-Thought and then the Final Answer in the exact format above.
```

Table 10: The prompt used for the Reasoner agent.

## E Example

Presented below is a comprehensive example demonstrating how MAC-Reasoner resolves a logic problem. The process begins with an LLM translating the original problem into a formal representation modeled as a solver-executable program. Subsequently, we extract structured symbolic information from this program. Based on this information, the LLM generates a natural language Logic-Augmented Context. Finally, the LLM performs logical reasoning leveraging this context to derive the solution.

**Logical Problem**

**Premises:**
If Adeline pursues her passion, then she will achieve success. Adeline is intelligent or dedicated. If Adeline sets goals, then she will work hard. If Adeline is dedicated, then she either has expertise or works hard (but not both). Adeline achieves success or has expertise. Adeline does not shape the opinions of others. Adeline sets goals. If Adeline achieves success, then she will gain recognition. If someone makes discoveries and publishes research, then they will have influence in their field. If Adeline is passionate, then she is dedicated and cares about others. Those who have influence in their field will shape the opinions of others. Briella sets goals. If someone gains recognition, then they will have influence in their field.
**Question:**
Based on the above information, is the following statement true, false, or uncertain? Adeline is intelligent.
**Options:**
A) True
B) False
C) Uncertain

Table 11: The Logical Reasoning Problem from ProverQA.

## Z3 Program

```
from z3 import *
# Variable declaration section
# Adeline's attributes and actions
adeline_pursues_passion = Bool('adeline_pursues_passion')
adeline_achieve_success = Bool('adeline_achieve_success')
adeline_intelligent = Bool('adeline_intelligent')
adeline_dedicated = Bool('adeline_dedicated')
adeline_sets_goals = Bool('adeline_sets_goals')
adeline_work_hard = Bool('adeline_work_hard')
adeline_has_expertise = Bool('adeline_has_expertise')
adeline_shapes_opinions = Bool('adeline_shapes_opinions')
adeline_gains_recognition = Bool('adeline_gains_recognition')
adeline_makes_discoveries = Bool('adeline_makes_discoveries')
adeline_publishes_research = Bool('adeline_publishes_research')
adeline_has_influence = Bool('adeline_has_influence')
adeline_passionate = Bool('adeline_passionate')
adeline_cares_about_others = Bool('adeline_cares_about_others')
# Briella's attributes and actions
briella_sets_goals = Bool('briella_sets_goals')
# Solver 1: Check if "Adeline is intelligent" is true
s1 = Solver()
# Constraint construction section
# [Adeline pursues her passion -> she will achieve success]
s1.add(Implies(adeline_pursues_passion, adeline_achieve_success))
# [Adeline is intelligent or dedicated]
s1.add(Or(adeline_intelligent, adeline_dedicated))
# [Adeline sets goals -> she will work hard]
s1.add(Implies(adeline_sets_goals, adeline_work_hard))
# [Adeline is dedicated -> she either has expertise or works hard (but not both)]
s1.add(Implies(adeline_dedicated, Xor(adeline_has_expertise, adeline_work_hard)))
# [Adeline achieves success or has expertise]
s1.add(Or(adeline_achieve_success, adeline_has_expertise))
# [Adeline does not shape the opinions of others]
s1.add(Not(adeline_shapes_opinions))
# [Adeline sets goals]
s1.add(adeline_sets_goals)
# [Adeline achieves success -> she will gain recognition]
s1.add(Implies(adeline_achieve_success, adeline_gains_recognition))
# [If someone makes discoveries and publishes research, then they will have influence in their field]
# (Applying this general rule to Adeline)
s1.add(Implies(And(adeline_makes_discoveries, adeline_publishes_research), adeline_has_influence))
# [If Adeline is passionate, then she is dedicated and cares about others]
s1.add(Implies(adeline_passionate, And(adeline_dedicated, adeline_cares_about_others)))
# [Those who have influence in their field will shape the opinions of others]
# (Applying this general rule to Adeline)
s1.add(Implies(adeline_has_influence, adeline_shapes_opinions))
# [Briella sets goals]
s1.add(briella_sets_goals)
# [If someone gains recognition, then they will have influence in their field]
# (Applying this general rule to Adeline)
s1.add(Implies(adeline_gains_recognition, adeline_has_influence))
# Option verification section
# Check if "Adeline is intelligent" can be True
s1.add(adeline_intelligent)
result_true = s1.check()
# Solver 2: Check if "Adeline is intelligent" is false
s2 = Solver()
# Constraint construction section (duplicate all constraints for separate solver)
# [Adeline pursues her passion -> she will achieve success]
s2.add(Implies(adeline_pursues_passion, adeline_achieve_success))
# [Adeline is intelligent or dedicated]
s2.add(Or(adeline_intelligent, adeline_dedicated))
# [Adeline sets goals -> she will work hard]
s2.add(Implies(adeline_sets_goals, adeline_work_hard))
# [Adeline is dedicated -> she either has expertise or works hard (but not both)]
s2.add(Implies(adeline_dedicated, Xor(adeline_has_expertise, adeline_work_hard)))
# [Adeline achieves success or has expertise]
s2.add(Or(adeline_achieve_success, adeline_has_expertise))
# [Adeline does not shape the opinions of others]
s2.add(Not(adeline_shapes_opinions))
# [Adeline sets goals]
s2.add(adeline_sets_goals)
# [Adeline achieves success -> she will gain recognition]
s2.add(Implies(adeline_achieve_success, adeline_gains_recognition))
# [If someone makes discoveries and publishes research, then they will have influence in their field]
# (Applying this general rule to Adeline)
s2.add(Implies(And(adeline_makes_discoveries, adeline_publishes_research), adeline_has_influence))
# [If Adeline is passionate, then she is dedicated and cares about others]
s2.add(Implies(adeline_passionate, And(adeline_dedicated, adeline_cares_about_others)))
# [Those who have influence in their field will shape the opinions of others]
# (Applying this general rule to Adeline)
s2.add(Implies(adeline_has_influence, adeline_shapes_opinions))
# [Briella sets goals]
s2.add(briella_sets_goals)
# [If someone gains recognition, then they will have influence in their field]
# (Applying this general rule to Adeline)
s2.add(Implies(adeline_gains_recognition, adeline_has_influence))
# Option verification section
# Check if "Adeline is intelligent" can be False
s2.add(Not(adeline_intelligent))
result_false = s2.check()
if result_true == sat and result_false == unsat:
    print("A")
elif result_true == unsat and result_false == sat:
    print("B")
else:
    print("C")
```

Table 12: Z3 solver program corresponding to the logical problem shown in Table 11.

**Symbolic Information**

```
--- Checking Solver: Solver#0 ---
[Result] sat
[Solution Constants]
adeline_pursues_passion = False
adeline_has_expertise = True
adeline_passionate = False
adeline_gains_recognition = False
adeline_dedicated = False
adeline_intelligent = True
adeline_makes_discoveries = False
adeline_shapes_opinions = False
adeline_achieve_success = False
adeline_work_hard = True
adeline_publishes_research = False
adeline_cares_about_others = False
briella_sets_goals = True
adeline_sets_goals = True
adeline_has_influence = False
--- Checking Solver: Solver#1 ---
[Result] unsat
[Unsatisfiable Core]
  [1] Or(adeline_intelligent, adeline_dedicated)
  [2] Implies(adeline_sets_goals, adeline_work_hard)
  [3] Implies(adeline_dedicated,
        Xor(adeline_has_expertise, adeline_work_hard))
  [4] Or(adeline_achieve_success, adeline_has_expertise)
  [5] Not(adeline_shapes_opinions)
  [6] adeline_sets_goals
  [7] Implies(adeline_achieve_success, adeline_gains_recognition)
  [8] Implies(adeline_has_influence, adeline_shapes_opinions)
  [9] Implies(adeline_gains_recognition, adeline_has_influence)
  [10] Not(adeline_intelligent)
```

Table 13: Symbolic information extracted from the Z3 solver program shown in Table 12.

## Logic-Augmented Context (Part I)

```
The logical problem asks whether the statement "Adeline is intelligent" is true, false, or uncertain based on the provided
      information.

The logic solver was used to evaluate two scenarios:
1.  Assuming "Adeline is intelligent" is True.
2.  Assuming "Adeline is intelligent" is False.

---

### Overall Answer

The statement "Adeline is intelligent" is **True**.

This is because assuming "Adeline is intelligent" leads to a consistent set of facts (SAT), while assuming "Adeline is
      intelligent" is false leads to a contradiction within the given constraints (UNSAT).

---

### Analysis per Option

#### Option A: "Adeline is intelligent" is True

**Feasibility:** This option is **feasible** (SAT). The solver found at least one assignment of logical variables that
      satisfies all given constraints, including `adeline_intelligent = True`.

**Concrete Feasible Solution:**

The following assignment of logical variables is consistent with all problem statements and `adeline_intelligent = True`:

* `adeline_pursues_passion` = False
* `adeline_achieve_success` = False
* `adeline_intelligent` = True
* `adeline_dedicated` = False
* `adeline_sets_goals` = True
* `adeline_work_hard` = True
* `adeline_has_expertise` = True
* `adeline_shapes_opinions` = False
* `adeline_gains_recognition` = False
* `adeline_makes_discoveries` = False
* `adeline_publishes_research` = False
* `adeline_has_influence` = False
* `adeline_passionate` = False
* `adeline_cares_about_others` = False
* `briella_sets_goals` = True

#### Option B: "Adeline is intelligent" is False

**Feasibility:** This option is **infeasible** (UNSAT). The solver determined that assuming `adeline_intelligent = False`
      creates a contradiction with the problem's core constraints.

**Explanation of Contradiction (UNSAT Core):**

The following set of constraints, when combined with the assumption `Not(adeline_intelligent)`, leads to a logical
      inconsistency:

1.  `Or(adeline_intelligent, adeline_dedicated)`: Adeline is intelligent or dedicated.
2.  `Implies(adeline_sets_goals, adeline_work_hard)`: If Adeline sets goals, then she will work hard.
3.  `Implies(adeline_dedicated, Xor(adeline_has_expertise, adeline_work_hard))`: If Adeline is dedicated, then she either
      has expertise or works hard (but not both).
4.  `Or(adeline_achieve_success, adeline_has_expertise)`: Adeline achieves success or has expertise.
5.  `Not(adeline_shapes_opinions)`: Adeline does not shape the opinions of others.
6.  `adeline_sets_goals`: Adeline sets goals.
7.  `Implies(adeline_achieve_success, adeline_gains_recognition)`: If Adeline achieves success, then she will gain
      recognition.
8.  `Implies(adeline_has_influence, adeline_shapes_opinions)`: Those who have influence in their field will shape the
      opinions of others (Adeline has influence -> Adeline shapes opinions).
9.  `Implies(adeline_gains_recognition, adeline_has_influence)`: If someone gains recognition, then they will have
      influence in their field (Adeline gains recognition -> Adeline has influence).
10. `Not(adeline_intelligent)`: **Assumption for this scenario** (Adeline is not intelligent).
```

Table 14: Part I of the Logic-Augmented Context: General analysis and feasibility check.

**Logic-Augmented Context (Part II)**

```
**Step-by-step contradiction:**

* Given assumption (10) `Not(adeline_intelligent)` and constraint (1) `Or(adeline_intelligent,
    adeline_dedicated)`, it implies `adeline_dedicated` must be True.
* Given constraint (6) `adeline_sets_goals` and constraint (2) `Implies(adeline_sets_goals,
    adeline_work_hard)`, it implies `adeline_work_hard` must be True.
* Given `adeline_dedicated = True` and `adeline_work_hard = True`, combined with constraint (3)
     `Implies(adeline_dedicated, Xor(adeline_has_expertise, adeline_work_hard))`, it forces `
    Xor(adeline_has_expertise, True)` to be True. This means `adeline_has_expertise` must be
    False (as `True XOR True` is False).
* Given `adeline_has_expertise = False` and constraint (4) `Or(adeline_achieve_success,
    adeline_has_expertise)`, it implies `adeline_achieve_success` must be True.
* Given `adeline_achieve_success = True` and constraint (7) `Implies(adeline_achieve_success,
    adeline_gains_recognition)`, it implies `adeline_gains_recognition` must be True.
* Given `adeline_gains_recognition = True` and constraint (9) `Implies(
    adeline_gains_recognition, adeline_has_influence)`, it implies `adeline_has_influence` must
     be True.
* Given `adeline_has_influence = True` and constraint (8) `Implies(adeline_has_influence,
    adeline_shapes_opinions)`, it implies `adeline_shapes_opinions` must be True.
* This conclusion directly contradicts constraint (5) `Not(adeline_shapes_opinions)`.

Therefore, the assumption that "Adeline is intelligent" is false leads to an unavoidable
    contradiction, proving this option infeasible.

---

### Variable State Reporting

#### Variables Involved in the Contradiction (Adeline is intelligent = False)

* `adeline_intelligent`: Forced to **False** (assumption), which initiates the contradiction
    chain.
* `adeline_dedicated`: Inferred **True** from `adeline_intelligent = False` and "Adeline is
    intelligent or dedicated".
* `adeline_sets_goals`: Explicitly stated as **True** in the problem.
* `adeline_work_hard`: Inferred **True** from `adeline_sets_goals = True` and "Adeline sets
    goals -> she will work hard".
* `adeline_has_expertise`: Inferred **False** from `adeline_dedicated = True`, `
    adeline_work_hard = True`, and "Adeline is dedicated -> she either has expertise or works
    hard (but not both)".
* `adeline_achieve_success`: Inferred **True** from `adeline_has_expertise = False` and "
    Adeline achieves success or has expertise".
* `adeline_gains_recognition`: Inferred **True** from `adeline_achieve_success = True` and "
    Adeline achieves success -> she will gain recognition".
* `adeline_has_influence`: Inferred **True** from `adeline_gains_recognition = True` and "
    Adeline gains recognition -> she will have influence in their field".
* `adeline_shapes_opinions`: Inferred **True** from `adeline_has_influence = True` and "Those
    who have influence in their field will shape the opinions of others". This directly **
    conflicts** with the explicit constraint "Adeline does not shape the opinions of others".
* Other variables (`adeline_pursues_passion`, `adeline_makes_discoveries`, `
    adeline_publishes_research`, `adeline_passionate`, `adeline_cares_about_others`, `
    briella_sets_goals`) are not part of this minimal conflicting set of constraints and
    therefore do not contribute to this specific contradiction.
```

Table 15: Part II of the Logic-Augmented Context: Detailed step-by-step contradiction derivation and variable state analysis.

**Reasoner**

```
Chain-of-Thought:
1.  Assume, for the sake of argument, that the statement "Adeline is intelligent" is False. This
     means that Adeline is not intelligent.
2.  The problem states, "Adeline is intelligent or dedicated." (Fact P2) Given our assumption
    from Step 1 that Adeline is not intelligent, it logically follows that Adeline must be
    dedicated.
3.  The problem explicitly states, "Adeline sets goals." (Fact P7) So, Adeline setting goals is
    true.
4.  The problem states, "If Adeline sets goals, then she will work hard." (Fact P3) Since we
    know from Step 3 that Adeline sets goals, it logically follows that Adeline will work hard.
5.  The problem states, "If Adeline is dedicated, then she either has expertise or works hard (
    but not both)." (Fact P4) We know from Step 2 that Adeline is dedicated. Therefore, it must
     be true that Adeline either has expertise or works hard (but not both).
6.  Combining the conclusion from Step 4 (Adeline works hard is true) with the conclusion from
    Step 5 (Adeline either has expertise or works hard, but not both), we can deduce the status
     of Adeline's expertise. For the "either/or but not both" condition to be true when "
    Adeline works hard" is true, it must be that "Adeline has expertise" is false. (If both
    were true, the "not both" condition would be violated; if both were false, the "either/or"
    condition would be violated). Thus, Adeline does not have expertise.
7.  The problem states, "Adeline achieves success or has expertise." (Fact P5) Given our
    conclusion from Step 6 that Adeline does not have expertise, it logically follows that
    Adeline achieves success.
8.  The problem states, "If Adeline achieves success, then she will gain recognition." (Fact P8)
     Since we know from Step 7 that Adeline achieves success, it logically follows that Adeline
     will gain recognition.
9.  The problem states, "If someone gains recognition, then they will have influence in their
    field." (Fact P13) Since we know from Step 8 that Adeline will gain recognition, it
    logically follows that Adeline will have influence in her field.
10. The problem states, "Those who have influence in their field will shape the opinions of
    others." (Fact P11) Since we know from Step 9 that Adeline will have influence in her field,
     it logically follows that Adeline will shape the opinions of others.
11. The problem explicitly states, "Adeline does not shape the opinions of others." (Fact P6)
12. Our conclusion in Step 10 (Adeline will shape the opinions of others) directly contradicts
    the explicit statement in Step 11 (Adeline does not shape the opinions of others). This
    contradiction arose from our initial assumption in Step 1 that Adeline is not intelligent.
13. Since assuming "Adeline is intelligent" is false leads to a contradiction, our initial
    assumption must be incorrect. Therefore, "Adeline is intelligent" must be True.

Final Answer: \boxed{A}
```

Table 16: Chain-of-Thought reasoning generated for the original logical problem based on the previously constructed Logic-Augmented Context.