

Do Large Language Models Excel in Complex Logical Reasoning with Formal Language?

Jin Jiang¹, Jianing Wang², Yuchen Yan³, Yang Liu²,
Jianhua Zhu¹, Mengdi Zhang², Liangcai Gao^{1*}

¹Wangxuan Institute of Computer Technology, Peking University, Beijing, China,

²Meituan Group, ³College of Computer Science and Technology, Zhejiang University

Correspondence: jiangjin@stu.pku.edu.cn, gaoliangcai@pku.edu.cn

Abstract

Large Language Models (LLMs) have been shown to achieve breakthrough performance on complex logical reasoning tasks. Nevertheless, most existing research focuses on employing formal language to guide LLMs to derive reliable reasoning paths, while systematic evaluations of these capabilities are still limited. In this paper, we aim to conduct a comprehensive evaluation of LLMs across various logical reasoning problems utilizing formal languages. From the perspective of three dimensions, i.e., spectrum of LLMs, taxonomy of tasks, and format of trajectories, our key findings are: 1) *Thinking* models significantly outperform *Instruct* models, especially when formal language is employed; 2) All LLMs exhibit limitations in inductive reasoning capability, irrespective of whether they use a formal language; 3) Data with PoT format achieves the best generalization performance across other languages. Additionally, we also curate the formal-relative training data to further enhance the small language models, and the experimental results ¹ indicate that a simple rejected fine-tuning method can better enable LLMs to generalize across formal languages and achieve the best overall performance.

1 Introduction

Logical reasoning, i.e., deductive, inductive, and abductive, is one of the imperative natural language processing (NLP) tasks and plays a significant role in artificial intelligence (AI) to perform human-like decision-making, task-solving, and deep-thinking (Zhang et al., 2021; Yang et al., 2023b; Yu et al., 2024; Xu et al., 2025). Different from conventional natural language understanding and generation, logical reasoning requires the AI systems to explicitly provide meticulous elucidation of thoughts and verifiable derivation chains,

^{*}Corresponding author.

¹Our codes and reports are available at <https://github.com/jiangjin1999/FormalEval>.

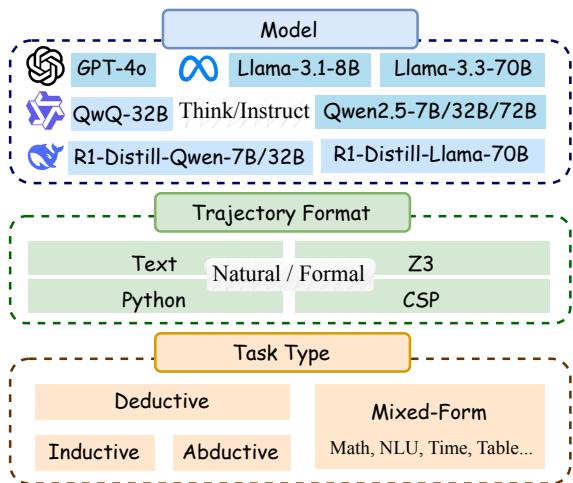


Figure 1: Evaluation framework with three specific dimensions: spectrum of LLMs, taxonomy of logical reasoning tasks, and format of output trajectories.

which is crucial and challenging (Cummins et al., 1991). Early works have developed multiple formal languages with symbol solvers to make the reasoning steps computable and structured (RANISE, 2003; Bulatov et al., 2005; Bjørner et al., 2015).

Recently, the emergence of reasoning capabilities in large language models (LLMs) has incentivized significant progress in complex reasoning tasks, such as mathematics, commonsense, and symbol (Achiam et al., 2023; Bi et al., 2024). Current studies have found that LLMs can achieve remarkable performance with the aid of formal language and symbol solvers, especially when integrating well-designed task-specific instructs (Lyu et al., 2023; Pan et al., 2023), chain-of-thought (CoT) reasoning patterns (Wei et al., 2022; Ye et al., 2023), and valuable solvers' feedback (He-Yueya et al., 2023; Gao et al., 2023; Wang et al., 2024). Such approaches aim to formalize the given logical problem and constantly adjust the results lean on the solver's feedback. Despite substantial efforts exhibiting exceptional performance, there are still

relatively limited systematic and comprehensive evaluations. Thus, a natural question remains open: *whether the LLM really excels in complex logical reasoning problems with formal language?*

To bridge the gap, this paper endeavors to perform a comprehensive evaluation of LLMs utilizing various formal languages to tackle diverse logical reasoning problems. At first, **we develop the evaluation architecture to clearly express the entire assessment view** (As illustrated in Section 2), with the framework shown in Figure 1. Specifically, we divide the entire assessment into three distinct dimensions, including the spectrum of LLMs, the taxonomy of logical reasoning tasks, and the format of trajectories. For the family of LLMs, we further consider different reasoning patterns which has been injected into the model training, such as short thinking (e.g., GPT-4o (Achiam et al., 2023), Qwen1.5/2/2.5 (Bai et al., 2023), LLaMA3/3.1/3.3 (Grattafiori et al., 2024)) and long thinking (e.g., DeepSeek-R1-Dsittill-Qwen (Guo et al., 2025)). For the logical reasoning, we adhere to the classic definitions (Flach and Kakas, 2000), categorizing tasks into deductive, inductive, and abductive reasoning. Additionally, we account for tasks that may integrate multiple reasoning types by introducing a new category referred to as mixed-form reasoning. Regarding the format of trajectories, we consider three main formal languages (“Python”, “Z3”, “CSP”) with a default natural language format as “Text”.

Secondly, we **perform a thorough evaluation across these three dimensions** (as detailed in Section 3). Many contemporary benchmarks purely emphasize informal text patterns and lack comprehensive integration of different formal languages and logical reasoning tasks (Lei et al., 2024; Xu et al., 2025; Xia et al., 2025). For instance, it is widely recognized that Python is superior to plain text when addressing mathematical problems (Friedman, 2023; Gao et al., 2023), but it remains unclear whether Python is equally effective in resolving BBH (Suzgun et al., 2022) and bbeh (Kazemi et al., 2025) problems. To fill this blank, this part aims to investigate whether current LLMs can solve a variety of logical reasoning tasks utilizing different formal languages. From this study, we derive several intriguing observations: 1) *Thinking* models significantly outperform *Instruct* models, especially when formal language is employed; 2) All LLMs exhibit limitations in inductive reasoning capability, irrespective of whether

they use a formal language; 3) LLMs typically produce inferior performance on difficult tasks. These findings prompt a new inquiry *Do large models possess generalization capabilities when employing formal languages?*

Thirdly, **we further investigate the generalization across different reasoning tasks and formal languages** (As illustrated in Section 4). To reach this goal, we collect a few training data from the training set of current evaluation tasks, which is classified into three types: deductive, inductive, and abductive. For each task type, we also provide different trajectories according to the usages of (in)formal languages. To make a fair comparison, we only use data from a single language type for SFT training, and the training data has the same scale size. From the experiments, we observe that the LLM can obtain significant in-domain performance on multiple logical reasoning tasks. In addition, we also discovered an elusive phenomenon that CSP is hard to generalize to other formal and informal languages, but it is easy to generalize from other languages to CSP. Therefore, we speculate that the poor performance of LLM on some formal languages can be blamed on the lack of pertinent knowledge and potential for stimulated reasoning.

Lastly, based on the previous exploration, **we aim to amplify the capabilities of weaker models in using formal languages to solve reasoning problems**. Concretely, we propose a simple but effective rejected fine-tuning (RFT) approach to curate different formal-relative training data. After the enrichment, the overall accuracy of using informal and formal languages for complex logical tasks can be improved by more than 10%.

In summary, the main contributions are as follows:

- In light of the insufficient evaluations of existing works, we aim to collect 66 tasks with multiple widely used formal languages, and provide a comprehensive evaluation for current LLMs across three dimensions, including the spectrum of LLMs, the taxonomy of tasks, and the format of trajectories.
- Considering that different formal languages have different expressions for reasoning, we explore the generalization across various formal languages.
- To further enhance the capability of LLMs in utilizing formal languages to solve complex

logic reasoning, we introduce a simple but effective rejected fine-tuning method with curated formal-relative data. The experimental results indicate the effectiveness of considering the generalization of formal language across various logical tasks.

2 Preliminary

As illustrated in Figure 1, our evaluation framework is structured along three dimensions: Model, Trajectory Format, and Task Type. In this section, we introduce the two key points of complex reasoning task categorization (Section 2.1) and trajectory format design (Section 2.2).

2.1 Taxonomy of Complex Logical Reasoning

Inspired by Xu et al. (2025), we present a unified taxonomy that categorizes a wide range of complex reasoning tasks into four major types: Deductive, Inductive, Abductive, and Mixed-Form. To elaborate, the categorization is based on the nature of reasoning required in human-like thinking in the real world: 1) **Deductive reasoning** is the forward reasoning process with rules that starts from the given premises to the conclusion (Goel, 2007; Johnson-Laird, 1999). Formally, we can denote the process as *premise* $\xrightarrow{\text{rule}}$ *conclusion*. 2) **Inductive reasoning** is the process that infers specific rules based on multiple premises and conclusions. It can be represented as $(\text{premise}, \text{conclusion}) \rightarrow \text{rule}$. 3) **Abductive reasoning** is the backward process of deductive which aims to obtain the premise based on conclusion, and the process can be viewed as *conclusion* $\xrightarrow{\text{rule}}$ *premise*. 4) **Mix-Form Reasoning** involves at least two of the above three types of reasoning. In real-life scenarios, most complex problems involve mixed reasoning, including but not limited to temporal-spatial reasoning, NLU, knowledge reasoning, and mathematical reasoning.

In pursuit of specific benchmarks based on these categories, we meticulously collect **66** subsets of data, and the detailed information can be found in Table 2. The details of the specific datasets are shown in Appendix A.

2.2 Trajectory Format

As shown in Figure 1, we categorize trajectory formats into two main types: **informal language** (natural language) and **formal language**. Informal language can be expressed as free-form text, while formal languages include programming languages

(e.g., Python) and logic-based languages (e.g., Z3 and CSP). They can be modeled as:

$$\mathcal{LLM}(Q) = \langle s_1, s_2, \dots, s_n \rangle \xrightarrow{\text{Exec}} A$$

where Q is the input question, and $\mathcal{LLM}(Q)$ represents the trajectory generated by LLM. Each step $s_i \in \mathcal{L}_{\text{LLM}}$ corresponds to a structured unit (e.g., code or logic expression), and the trajectory is executed by an external engine to produce the final answer A .

For **PoT**, we use Python 3.12 and its standard library as the execution environment. Each step $s_i \in \mathcal{L}_{\text{LLM}_{\text{PoT}}}$ is a valid Python statement. For **Z3**, we adopt the Z3 theorem prover as the executor and Z3 trajectories are composed of declarative symbolic steps $s_i \in \mathcal{L}_{\text{LLM}_{\text{Z3}}}$. For **CSP**, we use the python-constraint library as the trajectory executor. Each CSP trajectory $s_i \in \mathcal{L}_{\text{LLM}_{\text{CSP}}}$ consists of variable declarations, domain assignments, and constraint definitions.

In addition, we chose Z3 over Prover9 because Z3 not only supports first-order logic (Prover9-FOL) but also natively supports rich theories such as integers and arrays. More detailed description can be found in section B.

3 PART I: Evaluation across LLMs, Tasks, Trajectories

In PART I, we present a comprehensive evaluation across three dimensions: **Models**, **Trajectory Formats**, and **Reasoning Task Types**. Specifically, we evaluate both *Instruct* and *Thinking* models, ranging from 7B to 72B (see Figure 1). For reasoning tasks, we follow the taxonomy introduced in Section 2.1. For trajectory formats, we evaluate three formal languages and natural language, as detailed in Section 2.2. All evaluations are conducted in a zero-shot setting. For formal languages (PoT, Z3, CSP), we apply a three-step self-refinement process during code execution. Detailed evaluation settings are provided in Appendix C.1.

3.1 Model Performance for Reasoning Tasks and Trajectory Formats

As shown in Figure 2, the radar chart (Overall+Fine-grained) illustrates the model’s performance under different task types and trajectory formats. The complete results can be found in Appendix D.

Thinking model outperforms Instruct model

From the overall part, we can observe that series of

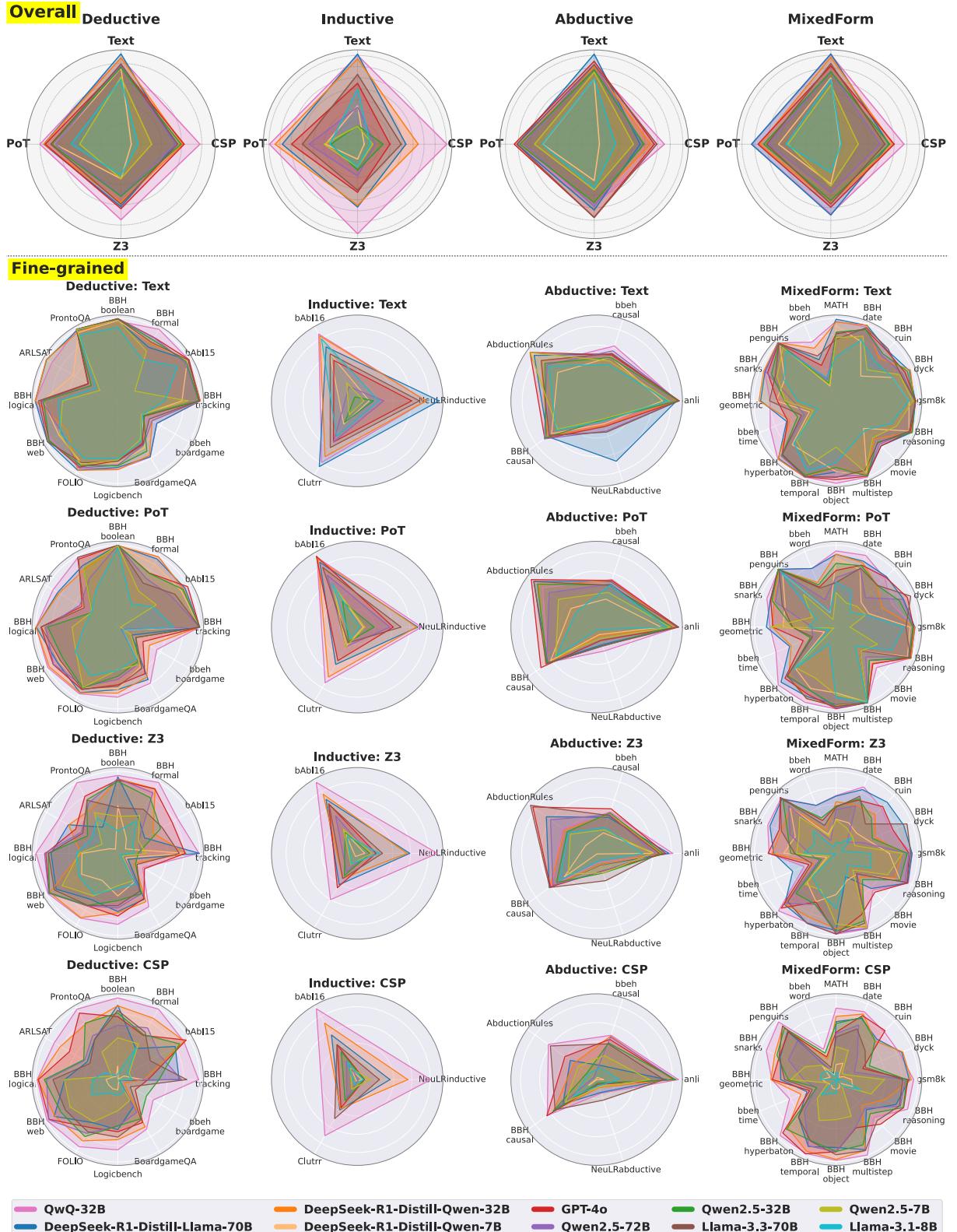


Figure 2: Radar plots illustrating the performance (%) of multiple LLMs across different reasoning task types (Deductive, Inductive, Abductive, Mixed Form) and trajectory formats (Text, PoT, Z3, CSP). Overall (top 1x4) shows aggregated performance by reasoning type and format. Fine-grained (below 4x4) present fine-grained results on individual tasks

Thinking models (e.g., QwQ-32B, etc.) outperform the *Instruct* series in most tasks, especially in the Inductive and Mixed-Form tasks. The disparities between them reflect that the *Thinking* mode can better elicit the LLM to provide reliable trajectories for formal reasoning. Previous evaluations (Xu et al., 2025) have demonstrated a similar finding that *Instruct* models have achieved unsatisfactory results in inductive reasoning, but they do not provide the suggestion that the *Thinking* model can perform well.

Text outperforms formal languages, except for

QwQ-32B Most models outperform formal languages in the Text trajectory format. In the Fine-grained section, as the trajectory format shifts from Text to CSP, the radar map coverage area gradually decreases, especially in the *bbeh* series of subtasks. However, QwQ-32B is the only model that stays ahead in all tasks and trajectories, maintaining a high level of performance in all formal languages.

Formal language performance drops significantly on difficult tasks

Models can achieve comparable or even better performance than Text with formal languages in simple tasks (e.g., Z3, CSP in Deducitive-*BBH_web*), but the performance of formal languages drops off substantially in complex tasks (e.g., Deductive-*bbeh_boardgameQA*). This phenomenon again suggests that current large models are better at using non-formal languages when expressing complex logic. Possible reasons include: 1) the model training process is dominated by natural language, with a scarcity of formal language samples; and 2) the model lacks augmentation for difficult and complex problems. The performance of text formatting is average, while formal language significantly decreases. It is worth noting that GPT-4o’s performance in this area is relatively stable, possibly due to its optimization in data.

Small models perform poorly on formal language

Both *Instruct* and *Thinking* small models have acceptable overall performance under Text, but when dealing with formal languages, the performance drops rapidly. Taking R1-Distill-Qwen-7B as an example, its performance under the CSP trajectory is even significantly lower than similar *Instruct* models, indicating that the *Thinking* mechanism is difficult to effectively support formal language reasoning at low parameter scales. In addition, in high complexity tasks such as *bbeh-time*,

bbeh-shuffle, etc., the small model is almost completely ineffective in structured trajectories such as Z3 and CSP, and it is difficult to complete the basic logical steps, which shows its serious lack of ability to deal with formal reasoning problems.

Overall, all models except QwQ-32B show a continuous performance degradation in the trajectory format change from Text to formal language (PoT, Z3, CSP). This phenomenon suggests that the current mainstream LLMs are more adept at handling natural language tasks, while they are still deficient in formal language reasoning.

3.2 Different Reasoning Tasks Prefer Different Trajectory Format

In this section, we use the GPT-4o result as an anchor point to conduct a detailed analysis of how different tasks exhibit varying preferences for trajectory formats. As shown in Figure 3, GPT-4o exhibits diverse preferences across trajectory formats. Below, we summarize the main observations.

Text performs better in language comprehension and open-ended tasks

First, in tasks such as *BBH_snarks*, *bbeh_linguini*, *bbeh_nycc*, Text is closer to the nature of the task in humor comprehension, linguistic style recognition, and fuzzy semantic parsing, and is superior to formal language. Secondly, in induction and abduction tasks such as *AbductionRules*, *NeuLRabductive*, *NeuLRInductive*, and *Clutrr*, where reasoning relies on linguistic expressions, the Text format is more advantageous. In addition, *LogicQA*, although categorized as a logic task, is more akin to a general knowledge quiz. It originates from the Chinese Civil Service Exam, where textual ability plays a dominant role in performance. (Cases in Figure 6)

Well-structured tasks prefer PoT

PoT format is particularly effective in tasks with strong structural characteristics, such as numerical computation and symbolic reasoning tasks like *BBH_dyck_languages* and *BBH_word_sorting*. In these settings, PoT enables efficient computation and facilitates the handling of rules involving nesting and ordering. Additionally, in tasks that involve temporal sequences, object tracking, and spatial reasoning, such as *bAbI16*, *bbeh_shuffled_objects*, and *bbeh_spatial_reasoning*, PoT demonstrates strong performance by leveraging programmatic trajectories to clearly express intermediate states and transformation processes. (Case in Figure 7)

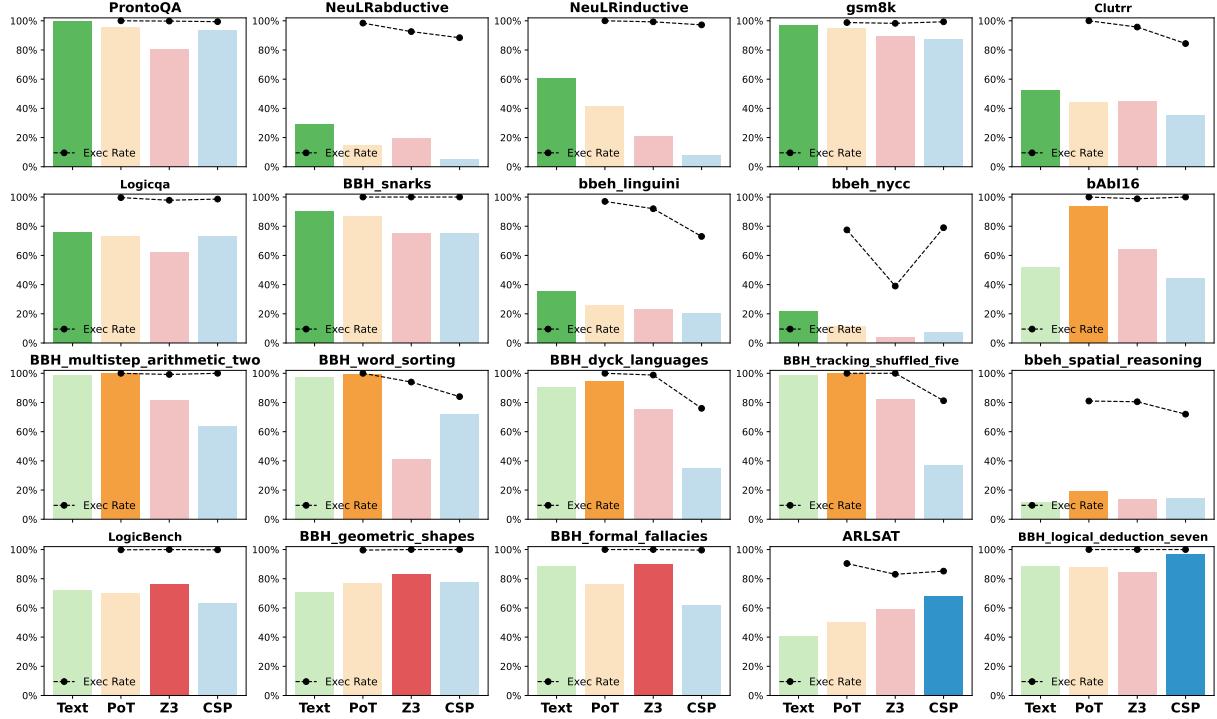


Figure 3: Preferred reasoning task performance across different trajectory formats (Text, PoT, Z3, CSP) in GPT-4o results. Each subplot shows task accuracy under different formats, with execution rate (Exec Rate) plotted as a black line. The highlighted bars represent the most preferred trajectory format for each task.

Z3 handles formal and FOL reasoning well. Z3 format shows a good adaptation to formal logic tasks, especially in tasks with strict logical rules: *LogicBench*, *BBH_formal_fallacies*, *BBH_logical_deduction*. This type of task is essentially convertible to first-order logical expressions, so using an SMT solver (e.g., Z3) as the trajectory language is more suitable. In addition, *BBH_geometric_shapes* involves spatial reasoning, where the boolean logical expressiveness of Z3 is more advantageous. (Case in Figure 9)

CSP shows advantages in complex constraints CSP format shows advantages in some structured logic tasks, such as *BBH_logical_deduction*, a result consistent with the findings of Logic-LM (Pan et al., 2023). More interestingly, in *ARLSAT*, a task derived from the Law School Admission Test, CSP also achieves the optimal result, which contrasts with the previous (Pan et al., 2023) literature’s conclusion that Z3 is better suited for this task. This difference may stem from the characteristics of the tasks themselves; in *ARLSAT*, the stems of the questions typically contain constraints, which are more consistent in form with the way CSPs are expressed. (Case in Figure 9)

Beyond the four dimensions mentioned above,

we can observe that execution success rate (Exec Rate) is also a key factor underlying the differences among various forms of language. Moreover, gsm8k achieves its best performance under the Text format, which is inconsistent with findings from previous studies (e.g., Ye et al. (2023); He-Yueya et al. (2023)). This discrepancy may be attributed to two factors: 1) Prior work often involves task-specific optimization for mathematical reasoning; 2) Current large language models are trained on substantial amounts of mathematical natural language reasoning data, which enhances their generalization ability in Text formats.

Overall, task trajectory alignment plays a critical role. Different tasks exhibit preferences for specific trajectory formats—some tasks are inherently better suited to certain formal representations, and using inappropriate formats may even hinder model performance. Therefore, when constructing multi-trajectory training or evaluation frameworks, it is important to carefully consider the alignment among task structure, target language, and model capabilities.

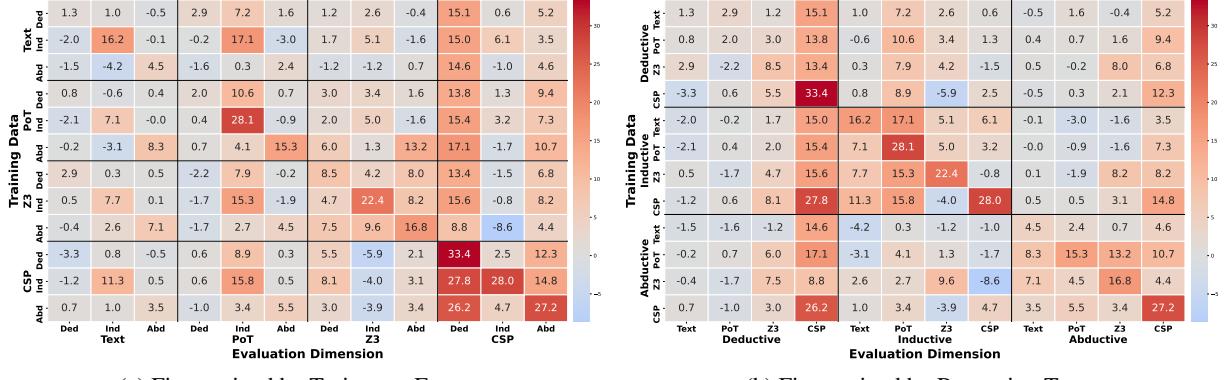


Figure 4: Generalization performance across **fine-grained** (task type \times format) configurations. Each cell shows the performance gain (Δ) from training on the row configuration and evaluating on the column configuration

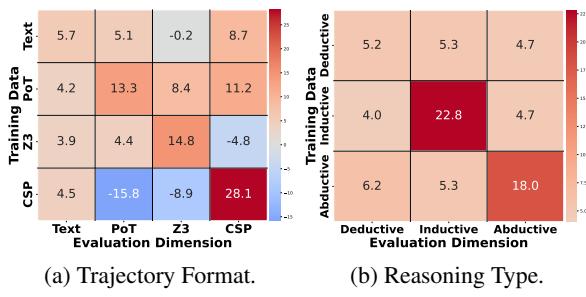


Figure 5: Generalization performance across reasoning types and trajectory formats (**coarse-grained** analysis). Each cell reports the performance gain (Δ) when training on the row group and evaluating on the column group.

4 PART II: Generalization Analysis across Reasoning Tasks and Trajectory Formats

4.1 Setup and Visualization Overview

We collected the training split of the evaluation dataset, categorized into three reasoning types: Deductive, Inductive, and Abductive (excluding Mixed-Form due to variable control challenges). Each training instance is represented in four trajectory formats: Text, PoT, Z3, and CSP. Details are provided in Section 5.1.

Based on the above data, we conduct two sets of analytical experiments: coarse-grained and fine-grained. 1): **Coarse-grained** experiments, as shown in Figure 5, involve training on 7 groups of data (3 reasoning types + 4 trajectory formats), each mixed with general-domain data, and evaluating on the same 3 reasoning types and 4 formats. 2): **Fine-grained** experiments, as shown in Figure 4, involve training on 12 groups of data (3 reasoning types \times 4 trajectory formats), each mixed

with general-domain data, and evaluating across all 12 combinations of reasoning types and formats. Each heatmap cell shows the performance gain (Δ) when training on the configuration in the row and evaluating on the configuration in the column. The performance gain reflects the improvement introduced by our constructed data when mixed with the general-domain data (Trained on Qwen-2.5-7B).

4.2 Coarse-Grained Generalization Analysis

Significant in-domain improvement The strongest performance gains are observed along the diagonal, indicating that the model benefits most when the training and evaluation data come from the same group. Notably, the improvements for CSP (Train) \rightarrow CSP (Eval) and Inductive (Train) \rightarrow Inductive (Eval) reach 28.1 and 22.8, respectively. Combined with observations from Part-I, this can be partially attributed to the relatively low baseline performance of the Qwen2.5-7B model on the CSP and Inductive dimensions, meaning that even a small amount of in-domain data leads to significant improvement.

PoT transfers well, while CSP transfers poorly Outside the diagonal, in figure 5a, PoT migrates well in Text, Z3, and CSP. This might be related to the fact that there is a lot of code data in the pre-training data. CSP, on the other hand, has an effect only on Text and CSP, with significant negative effects on PoT (-15.8) and Z3 (-8.9). This suggests that there may be structural differences among formal languages.

Reasoning types: all exhibit positive transfer The overall transfer effect is relatively balanced between the different reasoning types (Fig.5b). The relatively small improvement on Deductive itself

Model	Text		PoT		Z3		CSP		Avg	
	Acc	Acc	E-R	Acc	E-R	Acc	E-R	Acc	E-R	
GPT-4o	66.7	64.0	91.5	54.5	87.4	53.0	83.98	59.0	87.6	
Qwen2.5-7B-Instruct	52.3	37.0	78.6	33.0	70.0	25.0	52.1	37.0	66.9	
Qwen2.5-72B-Instruct	63.4	54.0	85.1	42.5	79.6	43.0	75.2	51.0	80.0	
Qwen2.5-7B-Baseline	49.7	40.0	75.4	27.1	68.2	20.0	52.2	34.0	65.3	
Qwen2.5-7B-Base w.Formal	52.7 ^{+3.0}	44.0 ^{+4.0}	83.5 ^{+8.1}	34.8 ^{+7.7}	76.5 ^{+8.3}	37.0 ^{+17.0}	68.1 ^{+15.9}	42.0 ^{+8.0}	76.0 ^{+10.7}	

Table 1: Performance of LLM on different trajectory formats before and after formal data enhancement. Accuracy (Acc) and execution rate (Exec Rate) are reported for text, PoT, Z3, and CSP formats. **Qwen2.5-7B-Baseline** denotes the baseline model trained with general data only; **Qwen2.5-7B-Base w.Formal** denotes the augmented model trained with a mixture of formal language data. Improvements after augmentation are shown in green.

may be related to the higher base level of the model on Deductive.

4.3 Fine-Grained Generalization Analysis

Deductive-CSP is most easily generalized In Figure 4, all entries in the Deductive-CSP column show improvements. The inclusion of any data contributes positively to its performance. This is mainly because CSP has a relatively low baseline, and the Deductive category contains some relatively simple tasks (*BBH_logical_deduction_three* from 40 % to 92%). As a result, adding similar data leads to performance gains.

CSP and Z3 transfer well across reasoning types

In Figure 4a, all entries (Ded/Ind/Abd) within the CSP and Z3 blocks show positive gains, indicating that regardless of reasoning type, CSP and Z3 formats can be effectively transferred.

Abductive transfers well across trajectory formats In Figure 4b, all entries (Text/PoT/CSP/Z3) within the Abductive block show improvements, suggesting that regardless of trajectory format, Abductive reasoning can be effectively transferred and improved.

5 PART III: Enhancing LLMs with Formal Data

5.1 Formal Data Construction via RFT

To enhance model capability in formal languages, we collect the portions of current evaluation datasets that overlap with training data as part of our training set. All dataset details are provided in Table 2. Similarly, the training data is categorized into three types: Deductive, Inductive, and Abductive, and four trajectory formats: Text, PoT, Z3, and CSP.

First, we extract up to 3,000 samples from all training data. Then, GPT-4o was chosen as the output for teacher model construction. In order to obtain high quality response data, we used **Rejection sampling Fine-Tuning (RFT)**. We used GPT-4o to sample the questions several times and then filtered out those samples whose code was executable and whose final answers are verified to be correct. The statistics of the filtered data are shown in Table 3. The number in parentheses after each model name indicates the amount of added data.

5.2 Main Result

As shown in Table 1, the enhanced model improves accuracy by 3.0% on Text, 7.7% on Z3 (with an 8.3% gain in execution rate), and 17.0% on CSP (from 20.0% to 37.0%, with a 15.9% increase in execution rate). Overall, average accuracy rises from 34.0% to 42.0%, and execution rate from 65.3% to 76.0%.

Beyond outperforming the baseline, our formal-data-enhanced model also surpasses the open-source model Qwen2.5-7B-Instruct across all formats. Qwen2.5-7B-Base w.Formal has a smaller parameter size than Qwen2.5-72B, but the performance gap is narrowed by formal data fine-tuning. This suggests that formal data augmentation can effectively improve the competitiveness of small models in formal reasoning tasks.

6 Related Work

6.1 Symbolic Solver Enhances LLM Reasoning

The integration of symbolic solvers with large language models (LLMs) has emerged as a promising approach to enhance logical reasoning. Early efforts focused on translating natural language to first-order logic (FOL), exemplified by the creation of

the MALLS dataset and the LogicLLaMA model, which demonstrated improved NL-to-FOL translation (Yang et al., 2023a). The Logic-LM framework further explored this direction by employing different formal languages and solvers tailored to specific reasoning tasks, such as FOL with Prover9, CSP solvers for constraint satisfaction, and Z3 for SMT problems (Pan et al., 2023). SATLM introduced declarative prompting to generate task specifications in logical formulas for SAT solvers (Ye et al., 2023), while LINC utilized LLMs for semantic parsing into FOL, offloading inference to theorem provers (Olausson et al., 2023). Subsequent research investigated strategies for improving NL-to-FOL translation through data generation and fine-tuning (Xiong et al., 2024), multi-step refinement of symbolic formulations (Wang et al., 2024), and the impact of pre-training data, including programming languages, on logical inference (Uchiyama et al., 2023). Frameworks like VERUS-LM aimed for versatility by supporting various reasoning tasks with a clear separation of knowledge and queries (Callewaert et al., 2025).

6.2 Complex Logical Reasoning Tasks

Evaluating the logical reasoning capabilities of LLMs necessitates challenging and diverse datasets that probe various aspects of inference. FOLIO, annotated with first-order logic, focuses on complex logical reasoning in natural language (Han et al., 2024). ProntoQA utilizes logic programming and emphasizes chain-of-thought reasoning (Saparov and He, 2023), while LogicBench covers propositional, first-order, and non-monotonic logic with a focus on single inference rules (Parmar et al., 2023). BOARDGAMEQA assesses reasoning with contradictory information and preferences (Kazemi et al., 2023), and AR-LSAT tests analytical reasoning skills using logic constraints (Zhong et al., 2022). The BIG-Bench Hard (BBH) benchmark includes a wide array of challenging tasks like Boolean Expressions (Suzgun et al., 2022), formal fallacies (Suzgun et al., 2022), logical deduction (Suzgun et al., 2022), shuffled objects (Suzgun et al., 2022), and web of lies (Suzgun et al., 2022). Other datasets like bAbI (Weston et al., 2015), CLUTRR (Sinha et al., 2019), α -NLI (Zhao et al., 2021), AbductiveRules (Bhagavatula et al., 2020), LogiQA (Liu et al., 2020), and gsm8k (Cobbe et al., 2021) target specific reasoning types such as deductive, inductive, abductive, temporal, spatial, and mathematical reasoning. The variety in these datasets and

their annotations highlights the multifaceted nature of complex reasoning and the ongoing efforts to evaluate and enhance LLMs in this domain.

7 Conclusion

In this paper, we provide a comprehensive evaluation of LLMs utilizing various formal languages to solve different categories of logical reasoning tasks. We first develop a systematic evaluation architecture and decompose it into three dimensions. Then, we perform a thorough evaluation across these three dimensions to show whether the current LLMs can excel in formal language utilization. Furthermore, we explore the generalization across multiple formal languages and provide a simple but effective method on the capability enhancement for small language models.

For future directions, on the one hand, we should strive to enhance the model’s reasoning capabilities in a balanced manner across different trajectory formats and task types, especially for Instruct models. At the same time, it may be valuable to construct formal language reasoning datasets in a "thinking" style. On the other hand, we can leverage the task-specific preferences for trajectory formats to further expand the capability boundaries of the model. One approach is to incorporate reasoning results from different trajectory formats as individual voters in a majority voting scheme. Another approach is to introduce multiple symbolic solvers for different reasoning trajectories during the thinking stage of the think model.

Limitations

This work provides a step toward evaluating and enhancing LLMs through formal reasoning formats, but several limitations remain. First, the landscape of LLMs is evolving rapidly. Our experiments focus on a limited set of models available at the time, and newer models may change performance trends. Second, while we include various reasoning types and benchmark datasets, the overall dataset coverage is limited. Our formal data augmentation is applied to a subset of tasks and may not generalize to other domains. Third, we focus on three formal formats, "PoT, Z3, and CSP," due to their executability and popularity. However, this excludes other symbolic systems such as Lean, Prolog, Coq, or SMT-LIB, which future work could explore. Finally, our formal data construction is based on the *Instruct* model (GPT-4o). With the rise of stronger *Think*-

ing models, generating think-style formal data may become more feasible and diverse in the future.

Acknowledgments

This work is supported by the projects of Beijing Nova Interdisciplinary Program (20240484647) and National Natural Science Foundation of China (No. 62376012), which is also a research achievement of State Key Laboratory of Multimedia Information Processing and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Rama Krishna Sai Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Yejin Choi, and Noah A Smith. 2020. Abductionrules: Training transformers to explain unexpected inputs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4246–4258.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, and 1 others. 2024. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*.

Nikolaj Bjørner, Anh-Dung Phan, and Lars Fleckenstein. 2015. *vz*-an optimizing smt solver. In *Tools and Algorithms for the Construction and Analysis of Systems: 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11–18, 2015, Proceedings* 21, pages 194–199. Springer.

Andrei Bulatov, Peter Jeavons, and Andrei Krokhin. 2005. Classifying the complexity of constraints using finite algebras. *SIAM journal on computing*, 34(3):720–742.

Benjamin Callewaert, Simon Vandevelde, and Joost Vennekens. 2025. Verus-lm: a versatile framework for combining llms with symbolic reasoning. *arXiv preprint arXiv:2501.14540*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. In *null*.

Denise D Cummins, Todd Lubart, Olaf Alksnis, and Robert Rist. 1991. Conditional reasoning and causation. *Memory & cognition*, 19:274–282.

Peter A Flach and Antonis C Kakas. 2000. *Abduction and Induction: Essays on their relation and integration*, volume 18. Springer Science & Business Media.

Robert Friedman. 2023. Large language models and logical reasoning. *Encyclopedia*, 3(2):687–697.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR.

Vinod Goel. 2007. Anatomy of deductive reasoning. *Trends in cognitive sciences*, 11(10):435–441.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shiron Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, and 1 others. 2024. Folio: Natural language reasoning with first-order logic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22017–22031.

Joy He-Yueya, Gabriel Poesia, Rose Wang, and Noah Goodman. 2023. Solving math word problems by combining language models with symbolic solvers. In *The 3rd Workshop on Mathematical Reasoning and AI at NeurIPS’23*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *Sort*, 2(4):0–6.

Philip N Johnson-Laird. 1999. Deductive reasoning. *Annual review of psychology*, 50(1):109–135.

Mehran Kazemi, Bahare Fatemi, Hritik Bansal, John Palowitch, Chrysovalantis Anastasiou, Sanket Vaibhav Mehta, Lalit K Jain, Virginia Aglietti, Disha Jindal, Peter Chen, and 1 others. 2025. Big-bench extra hard. *arXiv preprint arXiv:2502.19187*.

Mehran Kazemi, Quan Yuan, Deepti Bhatia, Najoung Kim, Xin Xu, Vaiva Imbrasaite, and Deepak Ramachandran. 2023. Boardgameqa: A dataset for natural language reasoning with contradictory information. *Advances in Neural Information Processing Systems*, 36:39052–39074.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Fangyu Lei, Qian Liu, Yiming Huang, Shizhu He, Jun Zhao, and Kang Liu. 2024. S3eval: A synthetic, scalable, systematic evaluation suite for large language model. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1259–1286.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2021. Logiqa: a challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 3622–3628.

Qiyuan Liu, Ming Yan, Yiyang Liu, Pan Lu, Siwei Wang, and Songfang Huang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1866–1877.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 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)*.

Theo X Olausson, Alex Gu, Benjamin Lipkin, Cede-gao E Zhang, Armando Solar-Lezama, Joshua B 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.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang 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.

Gaurav Parmar, Shikhar Murari, and Mohit Bansal. 2023. Logicbench: A challenging benchmark for logical reasoning with large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 148–168.

S RANISE. 2003. The smt-lib format: An initial proposal. *1st PDPAR, 2003*.

Abulhair Saparov and He He. 2023. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *The Eleventh International Conference on Learning Representations*.

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L Hamilton. 2019. Clutrr: A diagnostic benchmark for inductive reasoning from text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4506–4515.

Mirac Suzgun, Nathan Scales, Nathanael Scharli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Michael Petrov, Vincent Y Zhao, Ryan Murphy, Adam Roberts, and 1 others. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

Fumiya Uchiyama, Takeshi Kojima, Andrew Gambardella, Qi Cao, Yusuke Iwasawa, and Yutaka Matsuo. 2023. Which programming language and what features at pre-training stage affect downstream logical inference performance? *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1008–1021.

Marco Valentino, Mokanarangan Thayaparan, and André Freitas. 2022. Case-based abductive natural language inference. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1556–1568.

Xinyi Wang, Liangming Pan, and William Yang Wang. 2024. Logic-lm++: Multi-step refinement for symbolic formulations. In *The Twelfth International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Jason Weston, Antoine Bordes, Sumit Chopra, Tomas Mikolov, and Alexander Rush. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. In *null*.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart Van Merriënboer, Armand Joulin, and Tomas Mikolov. 2016. Towards ai-complete question answering: A set of prerequisite toy tasks. In *4th International Conference on Learning Representations, ICLR 2016*.

Yuan Xia, Akanksha Atrey, Fadoua Khmaissia, and Kedar S Namjoshi. 2025. Can large language models learn formal logic? a data-driven training and evaluation framework. *arXiv preprint arXiv:2504.20213*.

Siheng Xiong, Yuan Yang, Ali Payani, Ehsan Shareghi, and Faramarz Fekri. 2024. Strategies for improving nl-to-fol translation with llms: Data generation, incremental fine-tuning, and verification. *arXiv preprint arXiv:2409.16461*.

Fangzhi Xu, Qika Lin, Jiawei Han, Tianzhe Zhao, Jun Liu, and Erik Cambria. 2025. Are large language models really good logical reasoners? a comprehensive evaluation and beyond. *IEEE Transactions on Knowledge & Data Engineering*, pages 1–15.

Yuan Yang, Siheng Xiong, Ali Payani, Ehsan Shareghi, and Faramarz Fekri. 2023a. Harnessing the power of large language models for natural language to first-order logic translation. *arXiv preprint arXiv:2305.15541*.

Zonglin Yang, Xinya Du, Rui Mao, Jinjie Ni, and Erik Cambria. 2023b. Logical reasoning over natural language as knowledge representation: A survey. In *The 61st Annual Meeting Of The Association For Computational Linguistics*.

Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. 2023. Satlm: Satisfiability-aided language models using declarative prompting. In *Proceedings of NeurIPS*.

Nathan Young, Qiming Bao, Joshua Bensemman, and Michael J Witbrock. 2022. Abductionrules: Training transformers to explain unexpected inputs. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 218–227.

Fei Yu, Hongbo Zhang, Prayag Tiwari, and Benyou Wang. 2024. Natural language reasoning, a survey. *ACM Computing Surveys*, 56(12):1–39.

Jing Zhang, Bo Chen, Lingxi Zhang, Xirui Ke, and Haipeng Ding. 2021. Neural, symbolic and neural-symbolic reasoning on knowledge graphs. *AI Open*, 2:14–35.

Yixin Zhao, Adina Williams, Emily Dinan, Mohit Bansal, Mark Yatskar, and Yejin Choi. 2021. Adversarial nli: A new benchmark for natural language understanding. In *null*.

Victor Zhong, Chandra Bhagavatula, Ronan Le Bras, Yejin Choi, and Noah A Smith. 2022. Analytical reasoning of text: Unifying machine reading and logical reasoning. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2307–2323.

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. *arXiv e-prints*, pages arXiv–2104.

A Details of Datasets

Table 2 provides a comprehensive overview of all datasets used in our study. Each dataset is annotated with its reasoning type (Deductive, Inductive, Abductive, or Mixed-Form), along with the number of evaluation and training examples. We also include the original source for each dataset.

The classification follows our taxonomy introduced in Section 2.1. In particular:

- **Deductive** datasets include tasks that require formal logical reasoning based on explicit rules or premises.
- **Inductive** datasets focus on pattern discovery and generalization from limited examples.
- **Abductive** datasets involve generating plausible explanations under uncertainty.
- **Mixed-Form** includes tasks with hybrid or ambiguous reasoning types, further grouped into subcategories such as Temporal, NLU, Symbolic, Spatial, Knowledge, and Math.

Some datasets (e.g., BBH and bbeh) are split into finer task categories, each treated independently during evaluation. For large-scale datasets like GSM8K and MATH, we use a subset of examples (denoted by *) to maintain balance across task types.

This dataset collection forms the foundation for our evaluation across models, trajectory formats, and reasoning types.

B Detail of Trajectory Format

We extend the unified trajectory formulation to three specific formal languages: Python (PoT), Z3, and CSP. Each trajectory consists of a sequence of symbolic steps, which are executed by an external engine to compute the final answer.

We denote the model-generated trajectory as:

$$\mathcal{LLM}(Q) = \langle s_1, s_2, \dots, s_n \rangle \xrightarrow{\text{Exec}} A \quad (1)$$

Where Q is the input query, each s_i is a step in a domain-specific language, and A is the final answer produced by executing the trajectory.

Python (PoT) Trajectory

In the Python format, each step s_i is a syntactically valid Python statement. The trajectory consists of

Type	Dataset	Eval	Train	Original Source
Deductive	FOLIO	134	674	Han et al. (2024)
	ProntoQA	500	50818	Han et al. (2024)
	LogicBench	500	12908	Parman et al. (2023)
	BOARDGAMEQA	14K	750K	Kazemi et al. (2023)
	AR-LSAT	230	1629	Zhong et al. (2021)
	BBH (Boolean Expression)	250	-	Suzgun et al. (2022)
	bbeh (Boolean Expressions)	200	-	Kazemi et al. (2025)
	BBH (formal_fallacies)	250	-	Suzgun et al. (2022)
	bbeh (Zebra Puzzles)	200	-	Kazemi et al. (2025)
	BBH (logical_deductive_five_objects)	250	-	Suzgun et al. (2022)
	BBH (logical_deductive_seven_objects)	250	-	Suzgun et al. (2022)
	BBH (logical_deductive_three_objects)	250	-	Suzgun et al. (2022)
	bbeh (Boardgame QA)	200	-	Kazemi et al. (2025)
	BBH (tracking_shuffled_objects_five_objects)	250	-	Suzgun et al. (2022)
Inductive	BBH (tracking_shuffled_objects_seven_objects)	250	-	Suzgun et al. (2022)
	BBH (tracking_shuffled_objects_three_objects)	250	-	Suzgun et al. (2022)
	bbeh (Shuffled Objects)	200	-	Kazemi et al. (2025)
	BBH (web_of_lies)	250	-	Suzgun et al. (2022)
	bbeh (Web of Lies)	200	-	Kazemi et al. (2025)
	bAbI-15	1000	900	Weston et al. (2016)
	NeuLR-deductive	7001	-	Xu et al. (2025)
	CLUTRR	1042	2452	Sinha et al. (2019)
	bAbI-16	1000	900	Weston et al. (2016)
	NeuLR-inductive	7001	-	Xu et al. (2025)
Abductive	α -NLI	3059	169k	Valentino et al. (2022)
	AbductiveRules	2536	8848	Young et al. (2022)
	BBH (causal_judgement)	250	-	Suzgun et al. (2022)
	bbeh (Causal Understanding)	200	-	Kazemi et al. (2025)
	NeuLR-abductive	6001	-	Xu et al. (2025)
Mixed-Form				
NLU	Logical	LogiQA	1572	-
	Temporal	BBH (date_understanding)	250	-
		bbeh (Time Arithmetic)	200	-
		BBH (temporal_sequences)	250	-
		bbeh (Temporal Sequences)	200	-
	BBH (disambiguation_qa)	250	-	Suzgun et al. (2022)
	bbeh (Disambiguation QA)	200	-	Kazemi et al. (2025)
	BBH (hyperbaton)	250	-	Suzgun et al. (2022)
	bbeh (Hyperbaton)	200	-	Kazemi et al. (2025)
	BBH (ruin_names)	250	-	Suzgun et al. (2022)
Symbolic	bbeh (New Yorker Cartoon Caption)	200	-	Kazemi et al. (2025)
	BBH (salient_translation_error_detection)	250	-	Suzgun et al. (2022)
	bbeh (Linguini)	200	-	Kazemi et al. (2025)
	BBH (snarks)	250	-	Suzgun et al. (2022)
	bbeh (SARC Triples)	200	-	Kazemi et al. (2025)
	BBH (dyck_languages)	250	-	Suzgun et al. (2022)
	bbeh (Dyck Language)	200	-	Kazemi et al. (2025)
	BBH (word_sorting)	250	-	Suzgun et al. (2022)
	bbeh (Word Sorting)	200	-	Kazemi et al. (2025)
	BBH (geometric_shapes)	250	-	Suzgun et al. (2022)
Space	BBH (Geometric Shapes)	200	-	Kazemi et al. (2025)
	BBH (navigate)	250	-	Suzgun et al. (2022)
	bbeh (Spatial Reasoning)	200	-	Kazemi et al. (2025)
	BBH (penguins_in_a_table)	250	-	Suzgun et al. (2022)
	bbeh (Buggy Tables)	200	-	Kazemi et al. (2025)
Table	BBH (moive_recommendation)	250	-	Suzgun et al. (2022)
	bbeh (Movie Recommendation)	200	-	Kazemi et al. (2025)
	BBH (sports_understanding)	250	-	Suzgun et al. (2022)
	bbeh (SportQA)	200	-	Kazemi et al. (2025)
MATH	GSM8K	1319	*8790	Cobbé et al. (2021)
	MATH	5000	*7500	Hendrycks et al. (2021)
	BBH (multistep_arithmetaic_two)	250	-	Suzgun et al. (2022)
	bbeh (Multi-step Arithmetic)	200	-	Kazemi et al. (2025)
	BBH (object_counting)	250	-	Suzgun et al. (2022)
	bbeh (Object Counting)	200	-	Kazemi et al. (2025)
	BBH (reasoning_about_colored_objects)	250	-	Suzgun et al. (2022)
	bbeh (Object Properties)	250	-	Suzgun et al. (2022)

Table 2: Complex Logical Reasoning data categorization, data statistics, and sources.

variable assignments, arithmetic operations, control logic, and ends with a `print(A)` statement.

The Python trajectory is formalized as:

$$\mathcal{LLM}_{\text{Python}}(Q) = \quad (2)$$

$$\langle \text{stmt}_1, \text{stmt}_2, \dots, \text{stmt}_n, \quad (3)$$

$$\text{print}(A) \xrightarrow{\text{Python 3.12}} A \quad (4)$$

This trajectory is interpreted and executed sequentially using a Python 3.12 interpreter.

Z3 Trajectory

Inspired by Logic-LM (Pan et al., 2023), for Z3, the reasoning trajectory is constructed using the Z3 theorem prover. A typical trajectory includes symbolic variable declarations such as `x = Int('x')`, followed by logical assertions like `s.add(x > 1, x < 5)`, and ends with solver calls `s.check()` and `s.model()` to extract a result.

We represent the Z3 trajectory as:

$$\mathcal{LLM}_{\text{Z3}}(Q) = \quad (5)$$

$$\langle \text{Declare}, \text{Assert}_1, \dots, \text{Assert}_k, \quad (6)$$

$$\text{CheckSat}, \text{print}(A) \xrightarrow{\text{Z3 Solver}} A \quad (7)$$

Z3 supports a wide range of built-in logical theories, such as integer arithmetic, arrays, and bit-vectors.

CSP Trajectory

Constraint Satisfaction Problems (CSPs) are defined by a triple (X, D, C) , where $X = \{x_1, \dots, x_n\}$ denotes variables, $D = \{D_1, \dots, D_n\}$ their domains, and $C = \{C_1, \dots, C_m\}$ the set of constraints. Each constraint $C_j = \langle t_j, R_j \rangle$ is defined over a subset of variables and a relation on their domains.

The CSP trajectory is modeled as:

$$\mathcal{LLM}_{\text{CSP}}(Q) = \quad (8)$$

$$\langle \text{AddVar}_1, \dots, \text{AddVar}_n, \text{AddConst}_1, \dots,$$

$$\text{AddConst}_m, \text{GetSolution},$$

$$\text{print}(A) \xrightarrow{\text{python-constraint}} A \quad (9)$$

The execution uses the `python-constraint` solver. Variables are added through `addVariable()`, constraints through `addConstraint()`, and solutions are obtained via `getSolution()` or `getSolutions()`. The solver applies standard algorithms such as backtracking and constraint propagation.

While Prover9-FOL supports classical first-order logic, we choose Z3 for its broader practical applicability. Z3 not only supports FOL reasoning but also natively handles richer theories such as integers, arrays, and linear arithmetic. This allows it to express a wider range of constraints found in real-world reasoning tasks.

C Implementation Setups

C.1 Evaluations Details

In the inference phase, we use the vLLM (Kwon et al., 2023) framework for deployment. The inference configuration adopts greedy decoding strategy and sets the maximum generation length to 16K tokens. For the evaluation of model output, we adopt Qwen-2.5-72B-Instruct as the model evaluator to score.

C.2 Training Details

In terms of training implementation, we use Megatron-LM as the training framework with the following configurations: a cosine learning rate schedule is adopted with an initial learning rate of 1e-5, a warmup ratio of 0.03, and the learning rate decays to 0; the maximum sequence length is set to 8192, with a global batch size of 128, and the number of training epochs is set to 3. All experiments are completed with Supervised Fine-tuning (SFT) on a computing cluster consisting of 32 NVIDIA A100 GPUs.

D Complete results for different models

As shown in Table 3, we evaluated a total of 31 models across the three parts of this paper. Due to space constraints, we present the results of several representative models here: QwQ-32B (Table 6), GPT-4o (Table 7), Qwen2.5-7B (Table 8), and Qwen2.5-7B-Base w. Formal (Table 9). The complete results are provided in the supplementary files in Excel format.

E Supplementary Experiments for PART III

E.1 Data-Volume Alignment

In the main paper, QWEN2.5-7B-BASE w. FORMAL is obtained by continuing SFT on the same generic SFT corpus used by QWEN2.5-7B-BASELINE* and adding an extra pool of 28,060 formal-language instances. This raises a natural concern that the performance gain could partially

Section	Number	Model
PART-I		QwQ-32B
	4 Thinking-Model	DeepSeek-R1-Distill-Llama-70B
	+	DeepSeek-R1-Distill-Qwen-32B
	6 Instruct-Model	DeepSeek-R1-Distill-Qwen-7B
	=10	
		GPT-4o
		Qwen2.5-72B
		Qwen2.5-32B
		Llama-3.3-70B
		Qwen2.5-7B
PART-II		Llama-3.1-8B
	3 (Deductive, Inductive, Abductive)	Qwen2.5-7B-Base.w. Deductive (+5653)
	+	Qwen2.5-7B-Base.w. Inductive (+4947)
	4 (Text, PoT, Z3, CSP)	Qwen2.5-7B-Base.w. Abductive (+6557)
	=7	Qwen2.5-7B-Base.w. Text (+7384)
		Qwen2.5-7B-Base.w. PoT (+7448)
		Qwen2.5-7B-Base.w. Z3 (+6882)
		Qwen2.5-7B-Base.w. CSP (+6346)
		Qwen2.5-7B-Base.w. Deductive_Text (1376)
		Qwen2.5-7B-Base.w. Deductive_PoT (+1393)
		Qwen2.5-7B-Base.w. Deductive_Z3 (+1374)
		Qwen2.5-7B-Base.w. Deductive_CSP (+1510)
	3 (Deductive, Inductive, Abductive)	Qwen2.5-7B-Base.w. Inductive_Text (+1263)
	×	Qwen2.5-7B-Base.w. Inductive_PoT (+1476)
	4 (Text, PoT, Z3, CSP)	Qwen2.5-7B-Base.w. Inductive_Z3 (+1166)
	=12	Qwen2.5-7B-Base.w. Inductive_CSP (+1042)
PART-III	1 Baseline-Model+	Qwen2.5-7B-Base.w. Abductive_Text (+1820)
	1 Formal Data Enhanced Model	Qwen2.5-7B-Base.w. Abductive_PoT (+1775)
	=2	Qwen2.5-7B-Base.w. Abductive_Z3 (+1667)
		Qwen2.5-7B-Base.w. Abductive_CSP (+1295)
ALL	31	Qwen2.5-7B-Baseline* (15k) Qwen2.5-7B-Base.w. Formal (+28060)

Table 3: Comprehensive Overview of Model Evaluation Experiments in the Entire Paper (Models in Bold Are Presented with Full Results Later). Parentheses after the model in PART-II indicate the corresponding amount of data. All data are based on the 15k generic data of Qwen2.5-7B-Baseline*, plus (+) the corresponding amount of our synthetic data.

come from the larger total number of training examples. To isolate the effect of the *format* itself, we conducted a control where we *downsample* the training data of w. FORMAL to match the total size of the baseline (155,950 instances) via random sub-sampling.

Table 4 shows that (i) w. FORMAL still outperforms the baseline after size matching, and (ii) the absolute margins shrink slightly compared with the full-data setting, confirming that the improvements are primarily attributable to data *format* rather than data *volume*.

E.2 Results on a Different Base Architecture (Coder)

To further assess the robustness of formal-language supervision across base architectures, we repeat PART III on a *coder-oriented* backbone (QWEN2.5-CODER-7B). We report the instruct variant, our re-trained baseline*, and the format-augmented model (BASE.w. FORMAL). Table 5 shows consistent gains from formal trajectories across PoT/Z3/CSP, reflected both in accuracy and execution success rate, and leading to higher overall averages.

F Case Study

F.1 Case for PART I

We give cases where Text (Fig 6), PoT(Fig 7), Z3(Fig 8) and CSP(Fig 9) specialize in each case to show their strengths.

F.2 From Logic-LM Few-Shot Eval to Zero-Shot

Logic-LM uses few-shots setting and rule extraction to build “task-specific executable code” for “a particular formal language” and “a particular task”. We use zero-shot directly for evaluation. As shown in the case study in Figure 10, both approaches behave similarly and can evaluate the model’s formal language reasoning ability. Meanwhile, the zero-shot setting has better generalization, and this paper considers a subset of 66 datasets based on it.

G Prompts

For text, we use questions directly as input to the rubric. For formal languages, we use zero-shot reviews. Prompts are as follows: PoT in Figure 11; Z3 in Figure 12; CSP in Figure 13. The Prompt for evaluating models is in Figure 14.

Case Study for Text (bbeh_nycc)

The following is a description of a funny cartoon for the New Yorker Caption Contest

Description: Two trees are standing next to each other. One tree has a face and is looking at the other tree, which also has a face and is wearing a tie.

Which of the following captions is the funniest?

(A) 2 birds are doing it in my branches. Hence, the tie on the doorknob.
 (B) Next she's talking about my getting a trim and throwing out the squirrels.
 (C) It's part of my fall look.
 (D) It helps me feel more professional when I'm working from home.
 (E) She dresses me up, but we never go anywhere!
 (F) Sir, you can't come in here without a tie.
 (G) I just wanted to dress up one more time before I go bald again this fall.
 (H) Well it's just a seasonal position.
 (I) I got the job after the other guy got the axe.
 (J) Funny story, but it did start initially with a yellow ribbon

Case Study for Text (bbeh_linguini)

The plural of Catalan nouns is usually formed by adding the ending -s. But if the noun ends in one of the letters s, x or ç, more complex rules apply. Here are the singular and the plural forms of some Catalan nouns (in simplified spelling) and their English translations. Some forms are missing.

Singular | Plural | Translation\nel apèndix | els apèndixs | appendix
 el bastaix | (1) carrier\nel troleibús | (2) trolleybus
 el cactus | els cactus | cactus\nla càries | les càries | caries\n(3) | les clos | meadow
 el contumaç | els contumaços | rebel\nla faç | les façs | face
 el flux | els fluxos | stream\nel gimnàs | els gimnasos | gym
 la hèlix | les hèlixs | screw\nel index | els indexs | index
 el iris | (4) | rainbow\nel llaç | (5) | loop\nel ónix | (6) | onyx
 el pàncrees | els pàncrees | pancreas\nel pedaç | els pedaços | patch
 la pelvis | (7) | pelvis
 el permís | els permisos | permission\nel pis | (8) | flat, lodgings
 (9) | els sequaços | (male) follower
 (10) | les sequaçs | (female) follower\nel sufíx | els sufíxos | suffix
 (11) | els tastaolletes | frivolous person\nel teix | els teixos | yew
 la trencadís | les trencadís | piece of crockery
 el vas | els vasos | vase\nla xeix | (12) | (the letter) x

Fill in the masked element indicated with (5), assuming the (1), (2), ... elements are masked out elements.

Figure 6: Case for Text. 3.2. These issues are closely related to semantic understanding. They are better suited for textual representation and are difficult to address using formalized languages. Return to section 3.2.

Case Study for PoT (BBH_dyck_language)

Complete the rest of the sequence, making sure that the parentheses are closed properly.

Input: <(((<>))({{}[{}][<(>)}))

GPT-4o Text Answer:
 To correctly close a Dyck-
 n word, we need to ensure that all types of brackets
 are properly matched and closed in the correct order.
 The given sequence is:
 ...
 <(((<>))({{}[{}][<(>)}))
 ...

Let's break it down step by step:

1. '<' is opened and closed with '>'.
2. '{' is opened three times and needs to be closed three times.
3. '[' is opened twice and needs to be closed twice.
4. ']' is opened twice and needs to be closed twice.
5. '<' is opened and closed with '>'.
6. '{' is opened and closed with '}'.

The sequence is already correctly closed.
 Therefore, no additional characters are needed. The sequence is:
 ...
 <(((<>))({{}[{}][<(>)})) X

This sequence is a valid Dyck-
 n word as all brackets are properly matched and closed.

GPT-4o PoT Answer:

```
def complete_dyck_word(sequence):
    # Stack to keep track of opening brackets
    stack = []
    # Dictionary to match opening and closing brackets
    matching_bracket = {'(': ')', '{': '}', '[': ']', '<': '>'}

    # Iterate over the sequence
    for char in sequence:
        if char in matching_bracket:
            # If it's an opening bracket, push to stack
            stack.append(char)
        elif char in matching_bracket.values():
            # If it's a closing bracket, pop from stack
            if stack and matching_bracket[stack[-1]] == char:
                stack.pop()
            else:
                # If there's a mismatch, return an error
                return "Error: Mismatched brackets"

    # Complete the sequence by closing all remaining open brackets
    completed_sequence = sequence
    while stack:
        completed_sequence += matching_bracket[stack.pop()]

    return completed_sequence

# Input sequence
input_sequence = "<((<>))({{}[{}][<(>)}))"

result = complete_dyck_word(input_sequence)
print(result)
```

Output: <((<>))({{}[{}][<(>)}))> ✓

Figure 7: Case for PoT. Symbolic reasoning tasks are inherently well-suited to programming languages, and such problems may in fact originate from coding algorithm questions. Return to section 3.2

Case Study for Z3 (BBH_formal_fallacies)

"It is not always easy to see which chemicals are contained in our consumer products.
The following argument pertains to this question:
First, vitamin A is an ingredient of LIQUID EYELINER.
Second, every ingredient of BC Eye Cream 15 ml is not an ingredient of Mango Lip Butter or not an ingredient of LIQUID EYELINER.
Therefore, it is not the case that Vitamin A is an ingredient of BC Eye Cream 15 ml."
n

Is the argument, given the explicitly stated premises, deductively valid or invalid?

Options:
- valid
- invalid

GPT-4o Text Answer:

To determine whether the argument is deductively valid, we need to analyze the logical structure of the premises and the conclusion.

The argument is structured as follows:

1. Premise 1: Vitamin A is an ingredient of LIQUID EYELINER.
2. Premise 2: Every ingredient of BC Eye Cream 15 ml is not an ingredient of Mango Lip Butter or not an ingredient of LIQUID EYELINER.
3. Conclusion: Therefore, it is not the case that Vitamin A is an ingredient of BC Eye Cream 15 ml.

To assess the validity, we need to see if the conclusion necessarily follows from the premises.

Premise 2 can be restated in logical terms: For any ingredient (x) , if (x) is an ingredient of BC Eye Cream 15 ml, then (x) is not an ingredient of Mango Lip Butter or (x) is not an ingredient of LIQUID EYELINER.

Given Premise 1, Vitamin A is an ingredient of LIQUID EYELINER.

For the conclusion to be true, Vitamin A must not be an ingredient of BC Eye Cream 15 ml.

According to Premise 2, if Vitamin A were an ingredient of BC Eye Cream 15 ml, then it would have to be either not an ingredient of Mango Lip Butter or not an ingredient of LIQUID EYELINER.
Since we know from Premise 1 that Vitamin A is an ingredient of LIQUID EYELINER, the only way for Premise 2 to hold true is if Vitamin A is not an ingredient of BC Eye Cream 15 ml.

Thus, the conclusion logically follows from the premises, making the argument deductively valid.

- valid 

GPT-4o Z3 Answer:
from z3 import *

```
# Define the variables
VitaminA = Bool('VitaminA')
BC_Eye_Cream = Bool('BC_Eye_Cream')
Mango_Lip_Butter = Bool('Mango_Lip_Butter')
Liquid_Eyeliner = Bool('Liquid_Eyeliner')
```

Premises

```
# 1. Vitamin A is an ingredient of LIQUID EYELINER
premise1 = VitaminA == Liquid_Eyeliner
```

```
# 2. Every ingredient of BC Eye Cream 15 ml
```

```
# is not an ingredient of Mango Lip Butter or not
# an ingredient of LIQUID EYELINER
premise2 = Implies(BC_Eye_Cream,
\nOr(Not(Mango_Lip_Butter), Not(Liquid_Eyeliner)))
```

Conclusion

```
# It is not the case that Vitamin A is an ingredient of BC Eye Cream 15 ml
conclusion = Not(VitaminA == BC_Eye_Cream)
```

Check if the premises imply the conclusion

```
s = Solver()
```

```
s.add(premise1, premise2, Not(conclusion))
```

```
if s.check() == sat:
```

```
    print("invalid")
```

```
else:
```

```
    print("valid")
```

...The final answer is:

...

invalid 

Figure 8: Case for Z3. Z3 (which, in this context, incorporates the first-order logic reasoning capabilities of Prover9) excels at solving formal first-order logic problems. Return to section 3.2

Case Study for CSP (AR-LSAT)

Context:

On each of exactly seven consecutive days (day 1 though day 7), a pet shop features exactly one of three breeds of kitten—Himalayan, Manx, Siamese—and exactly one of three breeds of puppy—Greyhound, Newfoundland, Rottweiler. The following conditions must apply: Greyhounds are featured on day 1. No breed is featured on any two consecutive days. Any breed featured on day 1 is not featured on day 7. Himalayans are featured on exactly three days, but not on day 1. Rottweilers are not featured on day 7, nor on any day that features Himalayans.

Question:

If Himalayans are not featured on day 2, which one of the following could be true?

Options:

- A: Manx are featured on day 3.
- B: Siamese are featured on day 4.
- C: Rottweilers are featured on day 5.
- D: Himalayans are featured on day 6.
- E: Greyhounds are featured on day 7.

GPT-4o Text Answer:

To solve this problem, we need to analyze the constraints and determine which option could be true if Himalayans are not featured on day 2.

Given constraints:

- 1. Greyhounds are featured on day 1.
- 2. No breed is featured on any two consecutive days.
- 3. Any breed featured on day 1 is not featured on day 7.
- 4. Himalayans are featured on exactly three days, but not on day 1.
- 5. Rottweilers are not featured on day 7, nor on any day that features Himalayans.

Let's break down the possibilities:

Since Greyhounds are featured on day 1, they cannot be featured on day 7.

Himalayans are featured on exactly three days, but not on day 1 or day 2 (as per the question).

Rottweilers cannot be featured on any day that features Himalayans, and they cannot be featured on day 7.

Now, let's consider each option:

...

B: Siamese are featured on day 4.

This is possible. If Himalayans are on days 3, 5, and 6, Siamese can be on day 4.

C: Rottweilers are featured on day 5.

This is not possible because if Himalayans are on day 5, Rottweilers cannot be on the same day.

D: Himalayans are featured on day 6.

- This is possible. Himalayans can be on days 3, 5, and 6.

E: Greyhounds are featured on day 7.

- This is not possible because Greyhounds cannot be on day 7 as they are on day 1.

Therefore, the options that could be true are A, B, and D.

However, since the question asks for one option,

and considering the constraints,

option D: "Himalayans are featured on day 6"

is a straightforward possibility given the conditions.



GPT-4o CSP Answer:

```
from constraint import *

# Initialize the problem
problem = Problem()
# Define the variables for each day (1 to 7) for kittens and puppies
days = range(1, 8)
kitten_breeds = ['Himalayan', 'Manx', 'Siamese']
puppy_breeds = ['Greyhound', 'Newfoundland', 'Rottweiler']

# Add variables for kittens and puppies for each day
for day in days:
    problem.addVariable(f'kitten_day_{day}', kitten_breeds)
    problem.addVariable(f'puppy_day_{day}', puppy_breeds)

# Add constraints based on the problem statement
# Greyhounds are featured on day 1
problem.addConstraint(lambda p: p == 'Greyhound', ('puppy_day_1',))

# No breed is featured on any two consecutive days
for day in range(1, 7):
    problem.addConstraint(lambda k1, k2: k1 != k2, (f'kitten_day_{day}', f'kitten_day_{day+1}'))
    problem.addConstraint(lambda p1, p2: p1 != p2, (f'puppy_day_{day}', f'puppy_day_{day+1}'))

# Any breed featured on day 1 is not featured on day 7
problem.addConstraint(lambda k1, k7: k1 != k7, ('kitten_day_1', 'kitten_day_7'))
problem.addConstraint(lambda p1, p7: p1 != p7, ('puppy_day_1', 'puppy_day_7'))

# Himalayans are featured on exactly three days, but not on day 1
problem.addConstraint(lambda k: k != 'Himalayan', ('kitten_day_1',))

# Custom constraint to ensure Himalayans are featured on exactly three days
def himalayan_count(*args):
    return args.count('Himalayan') == 3

problem.addConstraint(himalayan_count,
    ('kitten_day_2', 'kitten_day_3', 'kitten_day_4', 'kitten_day_5', 'kitten_day_6', 'kitten_day_7'))

# Rottweilers are not featured on day 7, nor on any day that features Himalayans
problem.addConstraint(lambda p: p != 'Rottweiler', ('puppy_day_7',))
for day in days:
    problem.addConstraint(lambda k, p: not (k == 'Himalayan' and p == 'Rottweiler'),
        (f'kitten_day_{day}', f'puppy_day_{day}'))

# Additional condition: Himalayans are not featured on day 2
problem.addConstraint(lambda k: k != 'Himalayan', ('kitten_day_2',))

# Solve the problem and print the results
solutions = problem.getSolutions()

# Check which option could be true
for solution in solutions:
    if solution['kitten_day_3'] == 'Manx':
        print("A: Manx are featured on day 3.")
    if solution['kitten_day_4'] == 'Siamese':
        print("B: Siamese are featured on day 4.")
    if solution['puppy_day_5'] == 'Rottweiler':
        print("C: Rottweilers are featured on day 5.")
    if solution['kitten_day_6'] == 'Himalayan':
        print("D: Himalayans are featured on day 6.")
    if solution['puppy_day_7'] == 'Greyhound':
        print("E: Greyhounds are featured on day 7.")
```

Output: B: Siamese are featured on day 4.

Figure 9: Case for CSP. The problems in ARSAT involve numerous conditional constraints, which closely resemble the modeling approach used in Constraint Satisfaction Problems (CSPs). Return to section 3.2

Case Study for Few-Shot Formal-Eval in Logic-LM-->Our Zero-Shot Evaluation (AR-LSAT)

Context:

On each of exactly seven consecutive days (day 1 through day 7), a pet shop features exactly one of three breeds of kitten—Himalayan, Manx, Siamese—and exactly one of three breeds of puppy—Greyhound, Newfoundland, Rottweiler. The following conditions must apply: Greyhounds are featured on day 1. No breed is featured on any two consecutive days. Any breed featured on day 1 is not featured on day 7. Himalayans are featured on exactly three days, but not on day 1. Rottweilers are not featured on day 7, nor on any day that features Himalayans.

Question:

If Himalayans are not featured on day 2, which one of the following could be true?

Options:

- A: Manx are featured on day 3.
- B: Siamese are featured on day 4.
- C: Rottweilers are featured on day 5.
- D: Himalayans are featured on day 6.
- E: Greyhounds are featured on day 7.

A case from the original Logic-LM paper.

Logic-LM uses **few-shots** and **rule extraction** to build executable code for a “specific formal language” and a “specific task”.

We use zero-shot directly for evaluation.

As can be seen from the cases, both approaches behave similarly and both can effectively evaluate the formal language reasoning capability of the model. Meanwhile, zero-shot has better generalization, and this paper evaluates a subset of 66 data based on it.

Logic-LM

Task Description: You are given a problem description. The task is to parse the problem as a constraint satisfaction problem, defining the domain, variables, and constraints.

Context: A travel magazine has hired six interns – Farber, Gombarick, Hall, Jackson, Kanze, and Lha – to assist in covering three stories: Romania, Spain, and Tuscany. (... more context here ...) Jackson is assigned to Tuscany. Kanze is not assigned to Spain.

Question: Which one of the following interns CANNOT be assigned to Tuscany?

Options:

- (A) Farber
- (B) Gombarick
- (C) Hall
- (D) Kanze
- (E) Lha

Declarations:

```
stories = EnumSort([Romania, Spain, Tuscany])
assistants = EnumSort([photographer, writer])
(... more declarations here ...)
trained = Function([interns] -> [assistants])
```

Constraints:

```
trained(Gombarick) == trained(Lha) :::: Gombarick and Lha will be trained in the same field
trained(Farber) != trained(Kanze) :::: Farber and Kanze will be trained in different fields
(... more constraints here ...)
assigned(Jackson) == Tuscany :::: Jackson is assigned to Tuscany
assigned(Kanze) != Spain :::: Kanze is not assigned to Spain
```

Options:

```
is_unsat(assigned(Farber) == Tuscany) :::: (A)
is_unsat(assigned(Gombarick) == Tuscany) :::: (B)
is_unsat(assigned(Hall) == Tuscany) :::: (C)
is_unsat(assigned(Kanze) == Tuscany) :::: (D)
is_unsat(assigned(Lha) == Tuscany) :::: (E)
```

GPT-4o CSP Answer:

```
from constraint import *

# Initialize the problem
problem = Problem()
# Define the variables for each day (1 to 7) for kittens and puppies
days = range(1, 8)
kitten_breeds = ['Himalayan', 'Manx', 'Siamese']
puppy_breeds = ['Greyhound', 'Newfoundland', 'Rottweiler']

# Add variables for kittens and puppies for each day
for day in days:
    problem.addVariable(f'kitten_day_{day}', kitten_breeds)
    problem.addVariable(f'puppy_day_{day}', puppy_breeds)

# Add constraints based on the problem statement
# Greyhounds are featured on day 1
problem.addConstraint(lambda p: p == 'Greyhound', ('puppy_day_1',))

# No breed is featured on any two consecutive days
for day in range(1, 7):
    problem.addConstraint(lambda k1, k2: k1 != k2, (f'kitten_day_{day}', f'kitten_day_{day+1}'))
    problem.addConstraint(lambda p1, p2: p1 != p2, (f'puppy_day_{day}', f'puppy_day_{day+1}'))

# Any breed featured on day 1 is not featured on day 7
problem.addConstraint(lambda k1, k7: k1 != k7, ('kitten_day_1', 'kitten_day_7'))
problem.addConstraint(lambda p1, p7: p1 != p7, ('puppy_day_1', 'puppy_day_7'))

# Himalayans are featured on exactly three days, but not on day 1
problem.addConstraint(lambda k: k != 'Himalayan', ('kitten_day_1',))

# Custom constraint to ensure Himalayans are featured on exactly three days
def himalayan_count(*args):
    return args.count('Himalayan') == 3

problem.addConstraint(himalayan_count,
                      ('kitten_day_2', 'kitten_day_3', '\n' 'kitten_day_4', 'kitten_day_5', 'kitten_day_6', 'kitten_day_7'))

# Rottweilers are not featured on day 7, nor on any day that features Himalayans
problem.addConstraint(lambda p: p != 'Rottweiler', ('puppy_day_7',))
for day in days:
    problem.addConstraint(lambda k, p: not (k == 'Himalayan' and p == 'Rottweiler'),
                          (f'kitten_day_{day}', f'puppy_day_{day}'))

# Additional condition: Himalayans are not featured on day 2
problem.addConstraint(lambda k: k != 'Himalayan', ('kitten_day_2',))

# Solve the problem and print the results
solutions = problem.getSolutions()

# Check which option could be true
for solution in solutions:
    if solution['kitten_day_3'] == 'Manx':
        print("A: Manx are featured on day 3.")
    if solution['kitten_day_4'] == 'Siamese':
        print("B: Siamese are featured on day 4.")
    if solution['puppy_day_5'] == 'Rottweiler':
        print("C: Rottweilers are featured on day 5.")
    if solution['kitten_day_6'] == 'Himalayan':
        print("D: Himalayans are featured on day 6.")
    if solution['puppy_day_7'] == 'Greyhound':
        print("E: Greyhounds are featured on day 7.")
```

Figure 10: Case Study for Few-Shot Formal-Eval in Logic-LM—>Our Zero-Shot Evaluation (AR-LSAT). Logic-LM uses few-shots and rule extraction to build executable code for a “specific formal language” and a “specific task”. We use zero-shot directly for evaluation. As can be seen from the cases, both approaches behave similarly and both can effectively evaluate the formal language reasoning capability of the model. Meanwhile, zero-shot has better generalization, and this paper evaluates a subset of 66 data based on it.

Model	Text		PoT		Z3		CSP		AVG	
	ACC	ACC	Exec_Rate	ACC	Exec_Rate	ACC	Exec_Rate	ACC	Exec_Rate	
Qwen2.5-7B-Instruct	52.29	37.00	78.56	33.00	70.01	25.00	52.06	37.00	66.88	
Qwen2.5-7B-Baseline*	49.75	40.00	75.39	27.10	68.22	20.00	52.24	34.00	65.29	
Qwen2.5-7B-Base.w. Formal	52.67	44.00	83.48	34.81	76.53	37.00	68.09	42.00	76.03	
Qwen2.5-7B-Base.w. Formal _{downsampled}	51.18	45.00	82.48	32.63	72.20	35.00	65.31	41.00	73.33	

Table 4: Data-volume alignment (155,950 samples). The w. FORMAL advantage persists under equal training volume.

Model (Coder-7B)	Text		PoT		Z3		CSP		AVG	
	ACC	ACC	Exec_Rate	ACC	Exec_Rate	ACC	Exec_Rate	ACC	Exec_Rate	
Qwen2.5-Coder-7B-Instruct	48.84	39.00	81.40	24.99	64.30	18.00	44.86	33.00	63.52	
Qwen2.5-Coder-7B-Baseline*	47.07	41.00	79.91	28.90	71.48	20.00	50.06	34.00	67.15	
Qwen2.5-Coder-7B-Base.w. Formal	51.15	43.00	83.50	35.86	77.05	32.00	63.41	41.00	74.65	

Table 5: Supplementary PART III on coder-oriented base. Formal-language supervision consistently improves performance.

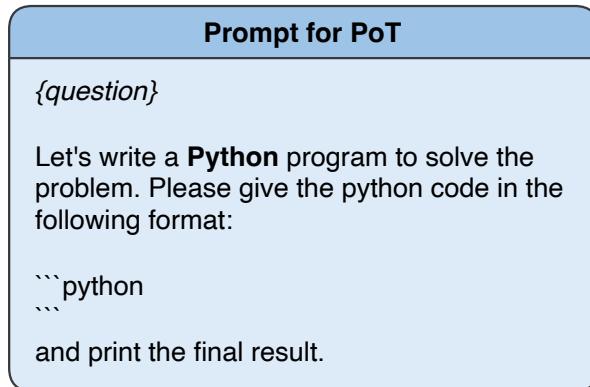


Figure 11: Prompt for PoT

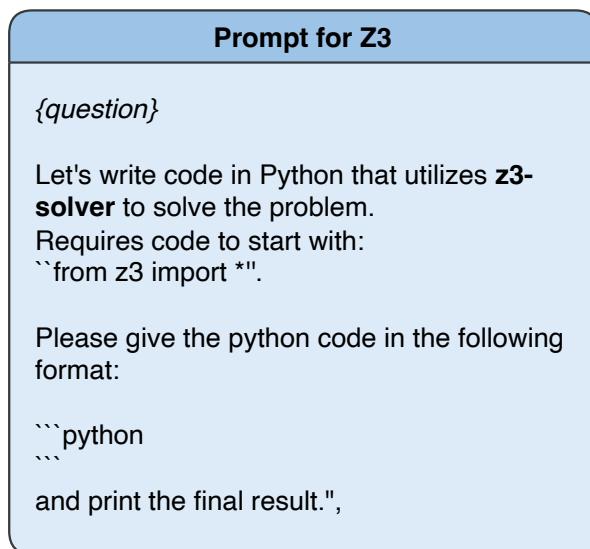


Figure 12: Prompt for Z3

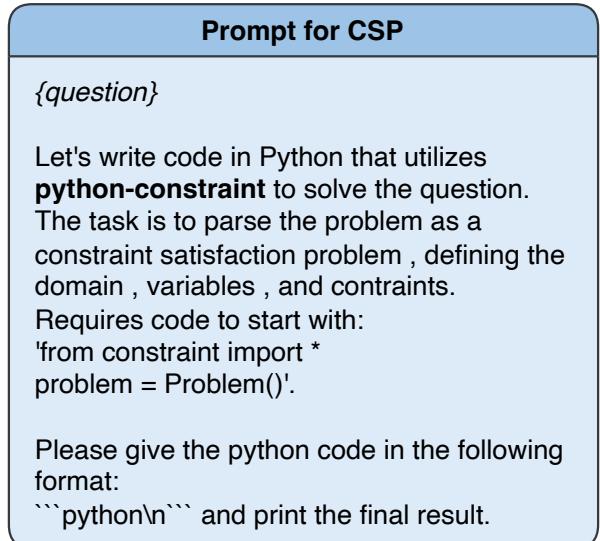


Figure 13: Prompt for CSP

Dataset	Text		PoT		Z3		CSP		AVG	
	ACC	ACC	Exec_Rate	ACC	Exec_Rate	ACC	Exec_Rate	ACC	Exec_Rate	
Average	75.0	68.6	85.1	61.9	79.4	65.1	82.2	67.6	82.2	
FOLIO	94.0	94.0	100.0	91.0	99.3	94.0	100.0	93.3	99.8	
ProntoQA	99.6	97.8	100.0	99.4	100.0	98.8	100.0	98.9	100.0	
logicbenchBQA	82.9	85.6	100.0	86.4	100.0	85.3	100.0	85.1	100.0	
BoardgameQA	78.5	79.3	99.9	75.1	100.0	69.6	100.0	75.6	100.0	
ARLSAT	92.2	91.3	100.0	83.0	97.0	89.1	100.0	88.9	99.0	
BBH_boolean_expressions	96.4	98.8	100.0	94.8	100.0	99.2	100.0	97.3	100.0	
bbeh_boolean_expressions	57.0	41.5	53.5	30.0	36.0	42.5	58.5	42.8	49.3	
BBH_formal_fallacies	100.0	99.2	100.0	99.6	99.6	98.8	100.0	99.4	99.9	
bbeh_zebra_puzzles	44.5	15.5	35.5	2.5	5.5	8.9	11.4	17.9	17.5	
BBH_logical_deduction_five_objects	100.0	100.0	100.0	99.6	100.0	98.0	100.0	99.4	100.0	
BBH_logical_deduction_seven_objects	99.2	99.6	100.0	100.0	100.0	100.0	100.0	99.7	100.0	
BBH_logical_deduction_three_objects	100.0	100.0	100.0	99.2	99.6	99.2	100.0	99.6	99.9	
bbeh_boardgame_qa	54.5	55.0	99.0	35.0	73.0	49.5	87.5	48.5	86.5	
BBH_tracking_shuffled_objects_five_objects	100.0	100.0	100.0	98.8	99.6	98.0	100.0	99.2	99.9	
BBH_tracking_shuffled_objects_seven_objects	100.0	100.0	100.0	96.8	100.0	99.2	100.0	99.0	100.0	
BBH_tracking_shuffled_objects_three_objects	100.0	100.0	100.0	100.0	100.0	99.2	100.0	99.8	100.0	
bbeh_shuffled_objects	41.5	0.5	2.0	3.5	10.0	3.0	9.5	12.1	7.2	
BBH_web_of_lies	92.8	98.8	100.0	98.0	99.6	99.2	100.0	97.2	99.9	
bbeh_web_of_lies	58.0	37.5	43.5	12.0	17.0	21.5	24.5	32.3	28.3	
bAbI15	99.3	92.8	100.0	84.1	98.1	92.6	99.9	92.2	99.3	
NeuLRdeductive	99.9	97.3	100.0	80.9	98.2	95.8	100.0	93.5	99.4	
clutr	78.8	73.3	100.0	60.1	94.2	71.0	98.7	70.8	97.6	
bAbI16	85.5	91.8	100.0	92.1	100.0	89.7	100.0	89.8	100.0	
NeuLRinductive	76.3	73.3	99.9	90.1	99.6	80.7	99.8	80.1	99.8	
anli	86.8	86.9	100.0	81.3	99.9	85.9	99.9	85.2	99.9	
AbductionRules	68.8	71.5	100.0	45.5	98.8	62.8	94.0	62.2	97.6	
BBH_causal_judgement	64.2	64.7	100.0	59.4	100.0	64.2	100.0	63.1	100.0	
bbeh_causal_understanding	62.0	53.5	99.5	46.5	90.5	49.0	94.5	52.8	94.8	
NeuLRabductive	26.0	26.9	99.9	9.9	95.7	15.1	94.1	19.5	96.6	
logicqa	86.5	82.9	100.0	77.9	99.6	80.5	99.9	82.0	99.8	
BBH_date_understanding	96.8	94.8	100.0	88.0	98.8	89.6	100.0	92.3	99.6	
bbeh_time_arithmetic	86.5	79.5	87.5	42.5	50.5	61.5	72.5	67.5	70.2	
BBH_temporal_sequences	100.0	99.6	100.0	91.6	99.2	97.2	99.2	97.1	99.5	
bbeh_temporal_sequence	52.5	0.0	0.5	0.0	0.0	1.0	1.5	13.4	0.7	
BBH_disambiguation_qa	48.0	54.0	100.0	38.8	100.0	46.4	100.0	46.8	100.0	
bbeh_disambiguation_qa	58.3	50.8	97.5	40.8	82.5	51.7	86.7	50.4	88.9	
BBH_hyperbaton	100.0	100.0	100.0	99.6	100.0	96.4	100.0	99.0	100.0	
bbeh_hyperbaton	38.0	26.5	56.5	19.0	35.0	18.0	52.0	25.4	47.8	
BBH_ruin_names	80.0	84.4	100.0	78.4	96.4	82.8	100.0	81.4	98.8	
bbeh_nycc	15.0	8.5	72.0	11.5	82.5	10.5	75.0	11.4	76.5	
BBH_salient_translation_error_detection	76.8	74.8	100.0	76.0	99.6	75.6	99.6	75.8	99.7	
bbeh_linguini	46.0	22.0	79.5	24.5	65.5	17.5	51.0	27.5	65.3	
BBH_snarks	93.8	91.6	98.9	90.5	100.0	92.1	100.0	92.0	99.6	
bbeh_sarc_triples	32.0	39.0	100.0	25.5	71.5	19.0	83.0	28.9	84.8	
BBH_dyck_languages	91.6	73.6	83.2	72.4	92.0	84.0	98.8	80.4	91.3	
bbeh_dyck_languages	49.0	30.0	79.5	22.0	69.0	28.0	67.5	32.3	72.0	
BBH_word_sorting	98.8	100.0	100.0	20.4	21.2	79.2	88.0	74.6	69.7	
bbeh_word_sorting	77.0	77.5	92.5	63.5	76.0	40.0	62.0	64.5	76.8	
BBH_geometric_shapes	80.0	80.8	100.0	82.0	99.2	78.4	100.0	80.3	99.7	
bbeh_geometric_shapes	40.5	9.0	19.5	12.5	25.5	14.5	32.0	19.1	25.7	
BBH_navigate	97.6	98.4	100.0	93.6	99.6	95.2	99.6	96.2	99.7	
bbeh_spatial_reasoning	43.0	40.5	53.0	32.5	48.0	30.5	47.5	36.6	49.5	
BBH_penguins_in_a_table	99.3	99.3	100.0	95.2	95.2	99.3	100.0	98.3	98.4	
bbeh_buggy_tables	25.0	16.5	28.5	8.0	16.0	3.5	15.0	13.3	19.8	
BBH_movie_recommendation	70.0	70.0	100.0	63.6	97.6	65.2	100.0	67.2	99.2	
bbeh_movie_recommendation	59.5	40.0	79.0	28.5	56.5	22.5	47.5	37.6	61.0	
BBH_sports_understanding	81.6	80.4	100.0	77.2	99.6	80.0	100.0	79.8	99.9	
bbeh_sportqa	53.0	15.0	23.0	17.0	34.0	17.5	34.5	25.6	30.5	
gsm8k	96.4	96.7	99.9	93.2	98.8	94.5	99.5	95.2	99.4	
MATH	96.5	93.1	99.4	70.7	88.3	87.1	99.0	86.9	95.6	
BBH_multistep_arithmetic_two	100.0	100.0	100.0	100.0	100.0	99.6	99.6	99.9	99.9	
bbeh_multistep_arithmetic	53.0	38.0	40.5	20.5	29.5	22.7	32.8	33.6	34.3	
BBH_object_counting	100.0	100.0	100.0	98.8	99.6	98.4	99.6	99.3	99.7	
bbeh_object_counting	58.0	31.0	85.5	40.5	71.0	22.5	61.0	38.0	72.5	
BBH_reasoning_about_colored_objects	100.0	98.8	100.0	95.6	99.2	94.4	99.2	97.2	99.5	
bbeh_object_properties	31.5	5.5	7.0	17.5	29.5	39.0	51.5	23.4	29.3	

Table 6: QwQ-32B Full Result.

Dataset	Text		PoT		Z3		CSP		AVG	
	ACC	ACC	Exec_Rate	ACC	Exec_Rate	ACC	Exec_Rate	ACC	Exec_Rate	
Average	66.7	63.5	91.5	54.5	87.4	52.8	84.0	59.4	87.6	
FOLIO	92.5	88.1	100.0	73.9	88.8	67.2	98.5	80.4	95.8	
ProntoQA	100.0	95.8	100.0	80.2	99.8	93.2	99.4	92.3	99.7	
logicbenchBQA	72.3	70.5	99.8	76.3	100.0	63.3	99.8	70.6	99.9	
BoardgameQA	59.7	66.0	100.0	63.5	97.9	60.2	98.1	62.4	98.7	
ARLSAT	40.9	50.4	90.4	59.1	83.0	67.8	85.2	54.6	86.2	
BBH_boolean_expressions	99.6	100.0	100.0	89.2	100.0	76.4	96.4	91.3	98.8	
bbeh_boolean_expressions	59.5	51.5	55.0	1.5	2.0	56.5	60.0	42.3	39.0	
BBH_formal_fallacies	88.4	76.4	100.0	90.4	100.0	62.0	99.6	79.3	99.9	
bbeh_zebra_puzzles	38.0	6.5	19.0	8.5	49.0	3.0	4.0	14.0	24.0	
BBH_logical_deduction_five_objects	93.2	94.0	99.6	87.6	99.6	96.4	100.0	92.8	99.7	
BBH_logical_deduction_seven_objects	88.8	88.0	100.0	84.8	100.0	96.8	100.0	89.6	100.0	
BBH_logical_deduction_three_objects	99.2	95.2	100.0	92.8	99.2	99.6	100.0	96.7	99.7	
bbeh_boardgame_qa	37.0	35.5	90.5	37.5	90.0	24.5	65.0	33.6	81.8	
BBH_tracking_shuffled_objects_five_objects	98.4	100.0	100.0	82.0	100.0	36.8	81.2	79.3	93.7	
BBH_tracking_shuffled_objects_seven_objects	100.0	99.6	100.0	82.4	100.0	41.6	75.2	80.9	91.7	
BBH_tracking_shuffled_objects_three_objects	100.0	100.0	100.0	55.6	100.0	53.6	74.4	77.3	91.5	
bbeh_shuffled_objects	29.5	59.0	83.5	36.0	77.5	23.5	49.0	37.0	70.0	
BBH_web_of_lies	96.4	91.2	100.0	96.4	100.0	96.4	100.0	95.1	100.0	
bbeh_web_of_lies	33.5	11.0	51.5	11.0	20.5	11.5	14.5	16.8	28.8	
bAbI15	99.6	98.7	100.0	76.2	97.9	95.9	99.8	92.6	99.2	
NeuLRdeductive	99.8	97.0	100.0	55.2	93.9	87.2	97.8	84.8	97.2	
clutrr	52.7	44.2	100.0	44.6	95.7	35.6	84.4	44.3	93.4	
bAbI16	51.8	93.4	100.0	64.4	98.8	44.1	100.0	63.4	99.6	
NeuLRinductive	60.3	41.2	100.0	21.1	99.3	7.9	97.2	32.6	98.8	
anli	88.8	87.6	100.0	73.4	99.9	81.6	100.0	82.9	100.0	
AbductionRules	88.5	86.6	100.0	84.3	100.0	41.2	63.8	75.2	87.9	
BBH_causal_judgement	69.0	73.8	100.0	61.0	100.0	64.7	100.0	67.1	100.0	
bbeh_causal_understanding	52.0	52.5	100.0	50.0	99.0	44.5	96.5	49.8	98.5	
NeuLRabductive	29.0	15.0	98.4	19.8	92.6	5.2	88.4	17.3	93.1	
logicqa	76.0	73.2	99.6	61.7	97.8	72.7	98.6	70.9	98.7	
BBH_date_understanding	94.0	82.0	100.0	70.4	98.8	84.4	100.0	82.7	99.6	
bbeh_time_arithmetic	63.5	43.5	74.0	36.0	66.0	35.0	73.5	44.5	71.2	
BBH_temporal_sequences	99.6	89.2	99.6	64.4	98.0	98.0	99.6	87.8	99.1	
bbeh_temporal_sequence	5.5	2.0	87.5	1.0	52.0	2.5	81.5	2.8	73.7	
BBH_disambiguation_qa	53.6	50.0	100.0	38.0	100.0	36.8	100.0	44.6	100.0	
bbeh_disambiguation_qa	63.3	54.2	98.3	44.2	96.7	70.8	93.3	58.1	96.1	
BBH_hyperbaton	92.8	88.4	100.0	94.8	99.6	92.0	99.2	92.0	99.6	
bbeh_hyperbaton	30.5	13.0	87.0	28.0	90.0	17.0	42.5	22.1	73.2	
BBH_ruin_names	86.4	86.0	100.0	80.8	98.4	84.4	99.2	84.4	99.2	
bbeh_nycc	21.5	11.5	77.5	3.5	39.0	7.0	79.0	10.9	65.2	
BBH_salient_translation_error_detection	73.2	80.0	100.0	84.4	100.0	76.8	100.0	78.6	100.0	
bbeh_linguini	35.0	25.5	97.0	23.0	92.0	20.0	73.0	25.9	87.3	
BBH_snarks	89.9	86.5	100.0	75.3	100.0	74.7	100.0	81.6	100.0	
bbeh_sarc_triples	30.0	37.5	97.5	17.5	45.5	35.5	85.0	30.1	76.0	
BBH_dyck_languages	90.4	94.4	100.0	75.2	98.8	35.2	76.0	73.8	91.6	
bbeh_dyck_languages	17.5	6.5	86.5	7.5	96.0	10.0	88.0	10.4	90.2	
BBH_word_sorting	97.6	99.2	100.0	41.2	94.0	71.6	84.0	77.4	92.7	
bbeh_word_sorting	46.5	53.0	89.5	43.5	83.5	30.0	61.5	43.3	78.2	
BBH_geometric_shapes	71.2	77.2	99.6	83.2	100.0	77.6	100.0	77.3	99.9	
bbeh_geometric_shapes	38.5	16.0	65.0	35.0	80.5	33.5	69.5	30.8	71.7	
BBH_navigate	99.2	98.0	100.0	78.4	100.0	74.4	98.0	87.5	99.3	
bbeh_spatial_reasoning	11.5	19.0	81.0	13.5	80.5	14.5	72.0	14.6	77.8	
BBH_penguins_in_a_table	98.6	100.0	100.0	96.6	99.3	82.2	97.3	94.4	98.9	
bbeh_buggy_tables	21.0	19.0	49.5	16.5	54.0	19.5	40.0	19.0	47.8	
BBH_movie_recommendation	77.2	63.6	100.0	67.2	99.2	76.8	99.6	71.2	99.6	
bbeh_movie_recommendation	60.5	26.0	99.5	34.0	85.0	27.5	85.0	37.0	89.8	
BBH_sports_understanding	86.8	87.6	100.0	54.4	100.0	60.8	100.0	72.4	100.0	
bbeh_sportqa	29.5	58.5	100.0	23.5	89.5	26.0	85.5	34.4	91.7	
gsm8k	96.7	94.6	98.8	89.3	98.3	87.6	99.3	92.1	98.8	
MATH	81.3	69.3	88.7	57.9	91.2	57.7	83.5	66.6	87.8	
BBH_multistep_arithmetic_two	98.8	100.0	100.0	81.2	99.2	63.6	100.0	85.9	99.7	
bbeh_multistep_arithmetic	27.0	1.5	23.0	1.0	13.5	1.0	16.0	7.6	17.5	
BBH_object_counting	95.6	99.6	100.0	98.8	100.0	88.0	100.0	95.5	100.0	
bbeh_object_counting	16.0	14.0	98.5	11.0	97.5	13.0	97.0	13.5	97.7	
BBH_reasoning_about_colored_objects	97.6	97.6	100.0	94.4	98.8	89.2	99.6	94.7	99.5	
bbeh_object_properties	10.5	5.5	50.5	9.5	52.5	1.0	38.5	6.6	47.2	

Table 7: GPT-4o Full Result.

Dataset	Text		PoT		Z3		CSP		AVG	
	ACC	ACC	Exec_Rate	ACC	Exec_Rate	ACC	Exec_Rate	ACC	Exec_Rate	
Average	52.3	36.9	78.6	33.0	70.0	24.8	52.1	36.7	66.9	
FOLIO	88.8	85.1	100.0	59.7	87.3	59.0	84.3	73.1	90.5	
ProntoQA	99.4	83.4	98.6	57.2	87.6	38.4	54.8	69.6	80.3	
logicbenchBQA	71.6	52.1	100.0	39.4	98.5	37.8	79.8	50.2	92.8	
BoardgameQA	54.3	52.8	99.1	38.2	89.5	29.2	73.8	43.6	87.5	
ARLSAT	25.2	36.5	93.9	22.2	51.3	8.7	22.6	23.2	55.9	
BBH_boolean_expressions	97.6	99.6	100.0	45.2	71.6	50.8	95.2	73.3	88.9	
bbeh_boolean_expressions	70.5	1.5	1.5	0.5	0.5	10.0	10.5	20.6	4.2	
BBH_formal_fallacies	69.6	50.8	100.0	52.8	91.6	50.0	88.4	55.8	93.3	
bbeh_zebra_puzzles	34.5	0.5	3.5	5.0	20.5	0.0	0.0	10.0	8.0	
BBH_logical_deduction_five_objects	66.8	56.0	100.0	46.8	69.2	64.8	95.2	58.6	88.1	
BBH_logical_deduction_seven_objects	66.0	55.2	100.0	47.2	67.6	70.8	94.8	59.8	87.5	
BBH_logical_deduction_three_objects	89.6	74.4	100.0	48.8	73.2	72.4	92.4	71.3	88.5	
bbeh_boardgame_qa	33.0	18.5	46.5	7.0	20.5	0.5	5.0	14.8	24.0	
BBH_tracking_shuffled_objects_five_objects	84.8	3.6	100.0	34.8	82.0	16.8	63.6	35.0	81.9	
BBH_tracking_shuffled_objects_seven_objects	85.2	5.2	100.0	43.2	82.0	15.6	68.4	37.3	83.5	
BBH_tracking_shuffled_objects_three_objects	89.2	0.4	100.0	35.6	76.4	22.0	58.4	36.8	78.3	
bbeh_shuffled_objects	59.5	4.0	26.5	2.0	12.5	1.5	4.0	16.8	14.3	
BBH_web_of_lies	81.2	59.2	100.0	78.4	94.4	66.8	74.8	71.4	89.7	
bbeh_web_of_lies	9.0	4.0	13.0	1.0	5.0	0.5	3.5	3.6	7.2	
bAbI15	23.7	54.3	99.9	29.1	90.7	16.0	64.2	30.8	84.9	
NeuLRdeductive	91.9	60.4	96.5	20.4	77.0	7.7	41.8	45.1	71.8	
clutrr	17.7	26.4	99.9	14.2	82.8	12.1	60.0	17.6	80.9	
bAbI16	23.7	55.8	99.9	31.3	91.6	14.8	63.5	31.4	85.0	
NeuLRinductive	7.4	8.8	96.5	16.1	91.1	14.4	53.3	11.7	80.3	
anli	77.7	78.8	99.8	59.8	95.1	55.6	83.7	68.0	92.9	
AbductionRules	88.3	50.6	81.4	34.8	41.8	23.9	37.0	49.4	53.4	
BBH_causal_judgement	51.9	54.0	100.0	37.4	92.5	40.6	85.6	46.0	92.7	
bbeh_causal_understanding	45.0	39.0	98.0	26.5	82.0	26.5	69.0	34.3	83.0	
NeuLRabductive	20.8	12.9	83.5	22.0	52.5	8.2	21.6	16.0	52.5	
logicqa	68.2	64.8	98.2	54.9	95.9	40.8	82.5	57.2	92.2	
BBH_date_understanding	84.8	32.0	100.0	33.6	73.6	38.4	84.8	47.2	86.1	
bbeh_time_arithmetic	30.5	3.0	32.0	9.0	52.5	10.5	46.0	13.3	43.5	
BBH_temporal_sequences	83.6	67.2	100.0	48.4	85.6	54.0	72.8	63.3	86.1	
bbeh_temporal_sequence	5.0	0.0	34.5	0.0	41.5	0.0	5.0	1.3	27.0	
BBH_disambiguation_qa	41.2	59.2	100.0	37.2	93.6	37.2	88.0	43.7	93.9	
bbeh_disambiguation_qa	45.8	29.2	90.0	35.8	81.7	15.8	44.2	31.7	71.9	
BBH_hyperbaton	68.0	70.0	100.0	53.6	94.4	32.4	57.6	56.0	84.0	
bbeh_hyperbaton	0.5	1.5	22.0	1.0	27.0	0.0	0.5	0.8	16.5	
BBH_ruin_names	53.2	36.0	100.0	28.4	82.8	10.4	39.2	32.0	74.0	
bbeh_nycc	10.5	8.5	94.0	7.0	59.0	4.5	41.5	7.6	64.8	
BBH_salient_translation_error_detection	47.2	10.0	100.0	30.4	58.0	21.6	60.0	27.3	72.7	
bbeh_linguini	18.0	16.0	81.5	6.0	73.5	4.5	52.0	11.1	69.0	
BBH_snarks	77.5	31.5	100.0	49.4	97.2	41.0	84.8	49.9	94.0	
bbeh_sarc_triples	16.0	12.5	91.5	12.0	74.5	5.5	22.5	11.5	62.8	
BBH_dyck_languages	83.2	39.2	100.0	24.8	76.8	14.8	40.0	40.5	72.3	
bbeh_dyck_languages	4.5	1.5	26.0	8.0	57.5	0.5	26.0	3.6	36.5	
BBH_word_sorting	32.4	97.2	100.0	61.2	86.0	6.0	32.4	49.2	72.8	
bbeh_word_sorting	21.0	10.0	58.5	23.0	76.0	10.5	27.5	16.1	54.0	
BBH_geometric_shapes	62.0	80.4	100.0	48.4	86.0	55.2	89.2	61.5	91.7	
bbeh_geometric_shapes	28.5	19.5	53.5	30.0	98.5	30.0	65.0	27.0	72.3	
BBH_navigate	84.0	50.4	100.0	43.6	86.0	29.6	50.8	51.9	78.9	
bbeh_spatial_reasoning	6.0	1.5	22.0	4.0	27.5	4.0	19.0	3.9	22.8	
BBH_penguins_in_a_table	91.1	56.2	100.0	63.0	78.8	24.7	48.6	58.7	75.8	
bbeh_buggy_tables	32.5	0.0	6.0	0.5	5.5	0.0	2.0	8.3	4.5	
BBH_movie_recommendation	63.2	42.0	100.0	28.8	72.4	10.0	42.0	36.0	71.5	
bbeh_movie_recommendation	33.0	23.5	83.5	26.5	84.0	3.5	16.0	21.6	61.2	
BBH_sports_understanding	74.0	58.0	100.0	55.6	99.2	48.4	95.6	59.0	98.3	
bbeh_sportqa	17.0	35.0	82.0	19.5	59.5	4.0	17.5	18.9	53.0	
gsm8k	93.0	17.1	21.3	81.1	96.5	58.9	81.3	62.5	66.4	
MATH	76.7	42.3	100.0	40.7	74.7	39.9	71.1	49.9	81.9	
BBH_multistep_arithmetic_two	94.0	98.8	100.0	98.4	99.6	48.4	82.0	84.9	93.9	
bbeh_multistep_arithmetic	6.5	0.0	0.0	0.5	7.0	0.0	3.0	1.8	3.3	
BBH_object_counting	56.0	82.8	100.0	86.4	95.6	48.8	71.2	68.5	88.9	
bbeh_object_counting	18.0	0.0	25.5	0.0	63.0	0.5	31.0	4.6	39.8	
BBH_reasoning_about_colored_objects	79.6	54.8	100.0	68.4	84.4	46.0	71.6	62.2	85.3	
bbeh_object_properties	21.0	0.5	25.0	2.0	37.0	0.0	22.5	5.9	28.2	

Table 8: Qwen2.5-7B Full Result.

Dataset	Text		PoT		Z3		CSP		AVG	
	ACC	ACC	Exec_Rate	ACC	Exec_Rate	ACC	Exec_Rate	ACC	Exec_Rate	
Average	52.7	43.9		83.5	34.8	76.5	37.0	68.1	42.1	76.0
FOLIO	83.6	78.4		99.3	71.6	100.0	61.2	97.8	73.7	99.0
ProntoQA	99.0	84.8		99.4	54.6	90.2	81.2	93.8	79.9	94.5
logicbenchBQA	80.1	70.4		100.0	72.3	99.1	64.2	99.6	71.8	99.6
BoardgameQA	68.8	50.0		94.9	50.9	97.8	46.5	92.3	54.1	95.0
ARLSAT	23.9	31.7		85.7	30.9	68.7	30.0	61.3	29.1	71.9
BBH_boolean_expressions	96.8	98.8		100.0	42.0	74.8	64.4	99.2	75.5	91.3
bbeh_boolean_expressions	70.0	0.0		53.0	3.0	4.0	22.5	33.0	23.9	30.0
BBH_formal_fallacies	68.0	58.4		100.0	61.2	100.0	54.8	98.0	60.6	99.3
bbeh_zebra_puzzles	39.0	12.5		46.5	13.5	39.5	8.0	22.0	18.3	36.0
BBH_logical_deduction_five_objects	61.6	57.6		100.0	62.8	94.0	77.6	96.8	64.9	96.9
BBH_logical_deduction_seven_objects	50.4	44.0		100.0	57.2	99.6	70.0	83.6	55.4	94.4
BBH_logical_deduction_three_objects	84.8	75.6		100.0	74.8	91.2	92.0	94.8	81.8	95.3
bbeh_boardgame_qa	31.5	0.0		84.5	12.5	37.0	18.0	46.0	15.5	55.8
BBH_tracking_shuffled_objects_five_objects	78.8	100.0		100.0	2.4	99.2	4.4	28.0	46.4	75.7
BBH_tracking_shuffled_objects_seven_objects	71.2	100.0		100.0	2.4	100.0	2.4	22.8	44.0	74.3
BBH_tracking_shuffled_objects_three_objects	80.8	99.6		100.0	8.0	100.0	5.2	16.8	48.4	72.3
bbeh_shuffled_objects	42.0	6.0		19.5	9.5	23.0	10.5	23.0	17.0	21.8
BBH_web_of_lies	85.6	63.2		100.0	77.2	100.0	58.0	64.0	71.0	88.0
bbeh_web_of_lies	12.5	13.0		51.5	1.0	9.5	2.5	7.0	7.3	22.7
bAbI15	99.1	96.1		97.3	89.6	99.1	89.2	98.5	93.5	98.3
NeuLRdeductive	92.1	85.7		99.0	37.1	84.9	49.8	86.7	66.2	90.2
clutrr	46.8	30.3		99.9	34.4	99.5	46.0	99.8	39.4	99.7
bAbI16	93.0	96.7		100.0	78.4	99.6	83.9	100.0	88.0	99.9
NeuLRinductive	8.2	8.6		100.0	18.0	98.6	14.1	93.8	12.2	97.5
anli	84.1	76.7		100.0	69.7	100.0	80.3	99.9	77.7	100.0
AbductionRules	91.0	85.7		99.8	47.7	95.0	48.3	81.1	68.2	92.0
BBH_causal_judgement	55.1	61.5		100.0	55.6	100.0	49.7	100.0	55.5	100.0
bbeh_causal_understanding	42.5	0.0		87.0	43.0	96.5	40.0	95.5	31.4	93.0
NeuLRabductive	12.0	15.4		93.8	17.3	84.3	1.0	27.4	11.4	68.5
logicqa	63.8	53.8		99.5	42.4	98.9	45.1	97.1	51.3	98.5
BBH_date_understanding	74.0	54.4		100.0	40.4	97.2	55.6	96.8	56.1	98.0
bbeh_time_arithmetic	32.0	10.5		59.5	7.5	43.5	10.5	45.5	15.1	49.5
BBH_temporal_sequences	30.0	42.8		98.8	33.6	98.4	36.0	76.0	35.6	91.1
bbeh_temporal_sequence	9.0	0.0		42.5	0.0	20.0	0.0	30.5	2.3	31.0
BBH_disambiguation_qa	35.6	46.0		100.0	41.2	99.2	64.0	99.2	46.7	99.5
bbeh_disambiguation_qa	42.5	0.0		43.3	34.2	90.0	61.7	100.0	34.6	77.8
BBH_hyperbaton	72.0	66.8		91.2	66.4	100.0	82.0	96.8	71.8	96.0
bbeh_hyperbaton	1.0	0.0		64.0	3.0	33.5	2.0	18.0	1.5	38.5
BBH_ruin_names	37.2	32.4		100.0	28.0	98.8	39.6	94.4	34.3	97.7
bbeh_nycc	8.0	0.0		62.5	1.5	95.5	5.0	72.5	3.6	76.8
BBH_salient_translation_error_detection	44.4	27.6		100.0	38.4	99.6	26.8	100.0	34.3	99.9
bbeh_linguini	18.0	0.0		53.5	2.5	79.0	10.5	63.0	7.8	65.2
BBH_snarks	73.6	50.6		100.0	51.1	98.9	41.6	100.0	54.2	99.6
bbeh_sarc_triples	16.5	12.0		96.5	16.5	43.0	34.0	100.0	19.8	79.8
BBH_dyck_languages	64.4	38.8		97.6	14.4	96.8	3.2	31.2	30.2	75.2
bbeh_dyck_languages	2.5	0.0		24.5	4.5	47.0	0.5	18.0	1.9	29.8
BBH_word_sorting	20.8	96.4		100.0	25.2	86.8	4.4	31.6	36.7	72.8
bbeh_word_sorting	20.5	12.5		73.5	6.5	84.5	1.5	15.0	10.3	57.7
BBH_geometric_shapes	53.6	13.6		100.0	29.2	99.2	60.8	99.2	39.3	99.5
bbeh_geometric_shapes	20.0	1.5		68.5	17.0	58.5	42.5	73.0	20.3	66.7
BBH_navigate	88.8	75.2		96.0	65.2	94.0	55.2	97.2	71.1	95.7
bbeh_spatial_reasoning	7.5	9.5		41.5	7.5	32.0	3.5	21.5	7.0	31.7
BBH_penguins_in_a_table	84.3	98.0		100.0	82.2	98.6	50.7	85.6	78.8	94.8
bbeh_buggy_tables	86.5	0.0		28.0	1.0	13.5	0.0	7.0	21.9	16.2
BBH_movie_recommendation	75.2	47.2		99.2	21.2	90.0	70.8	100.0	53.6	96.4
bbeh_movie_recommendation	49.0	0.0		52.5	11.0	71.0	9.0	74.5	17.3	66.0
BBH_sports_understanding	73.6	56.0		99.2	62.8	100.0	53.6	99.2	61.5	99.5
bbeh_sportqa	11.5	42.5		87.5	4.0	12.0	3.0	11.5	15.3	37.0
gsm8k	90.6	89.2		100.0	74.8	99.0	60.0	95.4	78.6	98.1
MATH	71.6	60.8		92.3	33.1	68.5	36.4	85.6	50.5	82.1
BBH_multistep_arithmetic_two	96.4	98.0		100.0	95.2	100.0	85.2	100.0	93.7	100.0
bbeh_multistep_arithmetic	10.5	0.0		60.5	0.0	2.0	0.0	7.0	2.6	23.2
BBH_object_counting	41.2	72.8		100.0	76.0	91.2	53.6	88.8	60.9	93.3
bbeh_object_counting	16.5	0.0		41.5	0.0	19.5	0.0	5.5	4.1	22.2
BBH_reasoning_about_colored_objects	71.6	87.2		100.0	60.4	96.0	59.2	90.8	69.6	95.6
bbeh_object_properties	9.5	1.5		25.0	1.0	10.0	0.0	4.0	3.0	13.0

Table 9: Qwen2.5-7B-Base.w.Formal Full result.

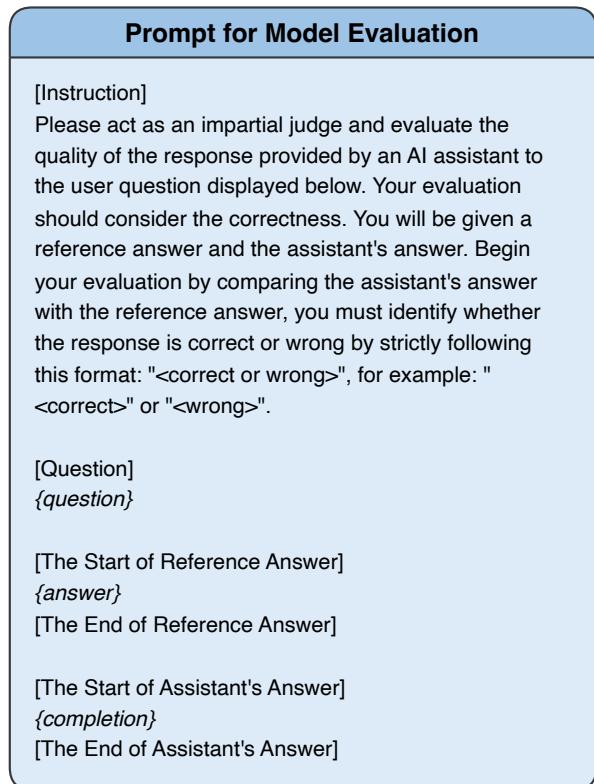


Figure 14: Prompt for Model Eval