FLoRE: A Formal Language Benchmark for Logical Reasoning Evaluation

Anonymous ACL submission

Abstract

Logical reasoning is a fundamental capability for advanced artificial intelligence systems. As Large Language Models(LLMs) continue to improve, many studies have been conducted to provide an accurate evaluation of LLM's logical reasoning capabilities. However, current benchmarks suffer from issues including interference from commensense-knowledge, short reasoning paths and low scalability. In this work, we propose an automated, cost-efficient method for generating dataset and FLoRE, a novel benchmark utilizing formal languages, a purely symbolic reasoning system, to evaluate the logical reasoning abilities of LLMs. Experimental results indicate that current large models generally perform poorly on logical reasoning tasks and are sensitive to the symbolic meanings involved in the reasoning process. This benchmark aims to leverage the characteristics of symbolic systems to avoid interference from commonsense knowledge, simultaneously maintaining the difficulty of reasoning tasks and reducing the complexity of data construction methods. All data and code will be available online.

1 Introduction

Logical reasoning is a cornerstone of artificial intelligence, enabling machines to simulate human thought processes and solve complex problems (Liu et al., 2025). Evaluating the **logical reasoning capabilities** of large language models (LLMs) is essential to ensure their reliability and trustworthiness in real-world applications (Deng et al., 2024). As LLMs are increasingly used in tasks that require complex decision-making, such as legal analysis, scientific research, and automated problem-solving (Gray et al., 2024; Nejjar et al., 2025; Aghakhani et al., 2025), it becomes critical to assess whether they can reason consistently, avoid contradictions, and draw valid conclusions from given premises.

Current methods for constructing evaluation datasets to assess the logical reasoning capabilities of LLMs primarily follow two paradigms: (1) manually designing rule-based templates and instantiating them with relevant content (Saparov et al., 2023; Yang et al., 2022; Han et al., 2022). (2) extracting logical reasoning patterns directly from source data. (Talmor et al., 2020; Zhong et al., 2021; Liu et al., 2023). However, existing datasets generally suffer from the following issues. First, current methods inevitably introduce **commonsense knowledge** (Bhargava and Ng, 2022), which interferes with the evaluation of the logical reasoning ability itself(e.g. north and south) (Evans et al., 1983; Chen et al., 2025; Parmar et al., 2024). To accurately assess logical reasoning, tests should focus on abstract or unfamiliar scenarios where answers must be derived through logical inference rather than retrieved from memory (Hua et al., 2024). Secondly, current logic reasoning benchmarks commonly feature **short reasoning paths**, which leads to insufficient reasoning difficulty (Patel et al., 2024). As deep-reasoning LLMs(such as Openai-o1 and Deepseek-r1) continue to improve, there is a growing trend of these benchmarks being gradually outperformed or surpassed. Moreover, many current benchmarks rely on manually crafted inference rules or a limited set of logical theorems for dataset construction, and the data collection process is challenging (Zhu et al., 2025), leading to **limited scalability**. Table 1 provides a comparison of our proposed dataset to datasets from previous studies.

To mitigate the influence of commonsense knowledge on reasoning and to explore a simple yet effective approach for dataset construction, we adopt formal language (Harrison, 1978) as the evaluation tool. In this paper, we examine whether pre-trained LLMs can perform satisfactory logical reasoning on formal languages. Based on a finite set of symbols, we design and generate four types

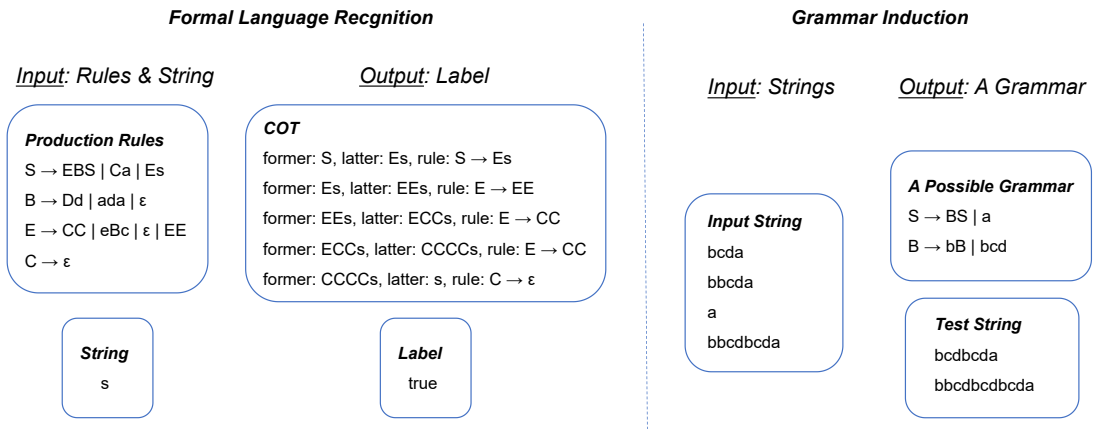

Figure 1: Examples of FLoRE. Based on formal language grammar, we design two logical reasoning tasks. In Formal Language Recognition task, we require the LLM to determine whether the *String* can be derived through inference based on the *Rules*. In grammar induction task, we provide several *Strings* as input, and require the LLM to output a *Possible Grammar* that can generate the input strings through derivation.

of seed grammars, and automatically construct a formal language grammar-string dataset. To our knowledge, it is the first logical reasoning benchmark utilizing the symbolic properties of formal language grammar reasoning to avoid interference from commensense knowledge. Using this dataset, we evaluate LLMs on both deductive and inductive reasoning tasks. We find that: (1) Current LLMs generally perform poorly on logical reasoning tasks based on Formal Language grammars. (2) LLMs are sensitive to the commonsense definitions of symbols; altering the definitions of these symbols can lead to significant performance degradation. (3) Guiding LLMs to explicitly articulate their reasoning processes can help improve their performance on reasoning tasks. Our contributions are as follows.

(1). We propose FLoRE, a novel benchmark designed to evaluate LLMs' logical reasoning abilities through formal language tasks. An example is shown in Figure 1.

(2). We develop a series of tools for the automatic analysis of results in formal grammar reasoning tasks and conduct a detailed analysis of the models' reasoning behavior.

(3). We evaluate the performances of several mainstream LLMs and find that current mainstream LLMs exhibit limited performance on tasks involving formal languages.

2 Related Work

2.1 Logic Reasoning Evaluation Benchmarks

Developing models capable of logical reasoning has been an important goal in the field of NLP since its inception (Cooper et al., 1996). Therefore, a number of studies have been conducted to measure the logical reasoning ability of LLMs (Xu et al., 2025). In current benchmarks, there are two common ways to build evaluation datasets. The **first** approach involves manually designing logical rules based on formal logic and then collecting data from various information sources to fill in the rule templates. PRONTOQA-OOD (Saparov et al., 2023) follows the deduction rules of *natural deduction* (Pfenning, 2004), and generates a proof that applies each rule. DEER (Yang et al., 2022) focuses on four common types of first-order logic (FOL) rules and constructs rule-fact pairs by collecting relevant facts via commercial search engines. FOLIO (Han et al., 2022) proposes a hybrid annotation method based on syllogisms, integrating machine-generated and expert-annotated data to construct datasets that conform to FOL. The **other** approach extracts content related to logical reasoning from existing texts or data resources and organizes it into datasets. LOT (Talmor et al., 2020) automatically generate data by sampling (subject, predicate, object) triples that are known to be either true or false from existing knowledge sources. AR-LSAT (Zhong et al., 2021) collects data from LSAT exams and selects questions from the analytical reasoning part to construct the dataset. LogiQA

Dataset	Commonsense Knowledge	Long reasoning path	Automatic generation	Low-cost expansion	Automatic evaluation
LogiQA	✓	×	×	×	✓
ProofWriter	✓	✓	✓	×	✓
FOLIO	✓	×	×	×	×
PRONTOQA	✓	×	✓	✓	✓
LogicBench	✓	×	✓	×	✓
Ours	×	✓	✓	✓	✓

Table 1: Comparison of existing logical reasoning datasets to our proposed benchmark. Previous work generally suffers from issues such as interference from commonsense interference, insufficient reasoning difficulty, and the complexity of data generation methods, which hinder the automatic and adaptive scaling of difficulty and problem quantity.

2.0 (Liu et al., 2023) extracts the premise and hypotheses from the concatenation of text, question, and options of cyber-downloaded machine reading comprehension instances to form natural language inference data.

However, existing evaluation benchmarks generally suffer from three main issues: interference from commonsense knowledge in logical reasoning (Evans et al., 1983), overly short reasoning paths (Patel et al., 2024), and high data construction costs that hinder scalability (Zhu et al., 2025). With FLoRE, we effectively address these challenges and conduct a thorough evaluation and analysis of the logical reasoning capabilities of LLMs.

3 Dataset Design and Construction

3.1 Notation

Formal language grammar (Jiang et al., 2009) is a system of rules used to generate and describe the structure of strings in a language. It plays a foundational role in both theoretical computer science and natural language processing (Harrison, 1978), enabling the analysis and generation of syntactically valid sequences. Here, we define a formal language grammar as a quadruple (Hopcroft et al., 2001):

$$G = (V, \Sigma, R, S) \quad (1)$$

In this definition, V represents the set of nonterminal symbols, which are used to denote syntactic categories or structures that can be further expanded through production rules. The second component, Σ, is the set of terminal symbols, which correspond to the actual symbols of the language and appear in the strings generated by the grammar. The third component, R, is the set of production rules, which describe how non-terminal symbols can be replaced by a combination of non-terminal and terminal symbols. Finally, S represents the start symbol, which is a specific non-terminal symbol from V that serves as the starting point for the derivation process. The common formats of formal language grammar are shown in Table 3.

Common tasks on formal language include **Formal Language Recognition** (Pecht, 1983) and **Grammar Induction** (De la Higuera, 2010). Formal Language Recognition relies primarily on **deductive reasoning** and pattern matching to determine string membership based on predefined rules (Fodor and Pylyshyn, 1988; Alfred et al., 2007), as the model must observe and apply possible grammatical rules step by step to determine whether a given string belongs to the language. To perform a Formal Language Recognition task, the process typically involves the following steps:

1). Observe the structure of the string and the grammar rules, and select the most likely rule at each step. 2). Apply the chosen rule to make substitutions and compare the current string with the target string to assess the difference. 3). If no suitable rule can be found, backtrack to a previous state and try a different path.

Grammar Induction requires **inductive reasoning** and hypothesis testing, as it requires the model to infer general grammatical rules from a limited set of example strings. Rather than applying predefined rules, the model must identify recurring patterns and structural regularities across the data, and generalize them into a coherent grammar (Gold, 1967; Ebner, 2021). To perform a Grammar Induction task, the process typically involves the following steps:

1). Observe the strings to identify commonalities and differences. 2). Break down the strings into substructures to extract underlying patterns. 3). Based on these patterns, define a set of symbols

and prototype production rules to form the initial grammar. 4). Test and refine the initial grammar, with details added to improve accuracy and completeness.

Formal languages can be categorized into different levels of complexity, most commonly described by the Chomsky hierarchy (Hunter, 2021). At the top of the hierarchy are recursively enumerable languages, which are the most expressive and not all are decidable (Sumitha and Geddam, 2011).

3.2 Task Definition

We design two logical reasoning tasks inspired by standard problems in formal language theory. One is **Formal Language Recognition(FLR)**, it requires the LLM to determine whether a given string can be derived from a specific formal language grammar. The task is formulated as a classification problem:

Let Σ be an alphabet, $L \subseteq \Sigma$ be a formal language defined over it. The problem of formal language recognition is to determine, for any given string $w \in \Sigma$, whether w belongs to the language L.

Formally, the LLM acts as a recognizer for L that implements a function:

$$\text{Recognize}_L : \Sigma^* \to \{0, 1\}$$

such that:

$$\text{Recognize}_L(w) = \begin{cases} 1, & \text{if } w \in L \\ 0, & \text{if } w \notin L \end{cases}$$

The other is **Grammar Induction(GI)**; it requires the LLM to infer a formal language grammar from a given set of strings:

Let Σ be an alphabet, and let $L \subseteq \Sigma^*$ be an unknown formal language. Consider two finite and disjoint subsets of strings:

1)The input set $S_{\text{in}} \subseteq \Sigma^*$, consisting of strings known to belong to the language L.

2)The test set $S_{\text{test}} \subseteq \Sigma^*$, consisting of strings used to evaluate the generalization ability of the induced grammar.

The grammar induction problem is to infer a formal grammar G such that:

1). All strings in the input set are derivable from G: $S_{\text{in}} \subseteq L(G)$

2). The grammar G maximizes coverage over the test set, i.e., the number of strings in S_test that are derivable from G is as large as possible: $\max |S_{\text{test}} \cap L(G)|$

Seed Grammar Type	Examples
`Root`	$S \to BCD$
`Context-Free`	$C \to Ab$
`Context-Sensitive`	$bSB \to a$
`Recursive`	$D \to DD$
`Epsilon`	$A \to \varepsilon$

Table 2: Examples of seeds we generated. Root refers to those whose derivations uniformly begin with a designated root symbol on the left-hand side; Context-Free has a non-recursive rule with a single non-terminal on the left-hand side; Context-Sensitive allows context-dependent rules where the output may be shorter than the input; Recursive includes a self-referential rule enabling unbounded derivation; Epsilon consists of productions that derive the empty string.

Previous studies have shown that if only positive examples are provided, most types of formal language grammar cannot be uniquely defined from these examples alone (Gold, 1967). Therefore, we define this task as an open-ended generation task with no single correct answer. The task is considered successfully completed as long as the LLM outputs a grammar that can generate all the given strings through valid derivations.

3.3 Data Construction

We propose a simple-yet-effective method for generating evaluation data, consisting of four main stages: Seed Generation, Group Mixing, Reliable Inference, and Random Perturbation. The complete process of dataset construction is illustrated in Figure 2. Currently, FLoRE consists of approximately 30.5k instances. Thanks to our generation methodology, the dataset can be easily scaled to a significantly larger size without compromising quality or consistency.

Seed Generation. We first generate a set of seed rules based on a predefined set of symbols, as the foundation for dataset construction.

Considering the performance of existing commercial LLMs on grammar-based reasoning tasks(shown in Appendix B), as well as the structural characteristics and hierarchical classification of formal language, we construct four types of seed grammar rules: Root, Context-Free, Context-Sensitive, Recursive, and Epsilon. Examples of each type of seed are shown in Table 2.

Group Mixing. Next, we conduct a mix-up on generated seed grammar, thus forming a number

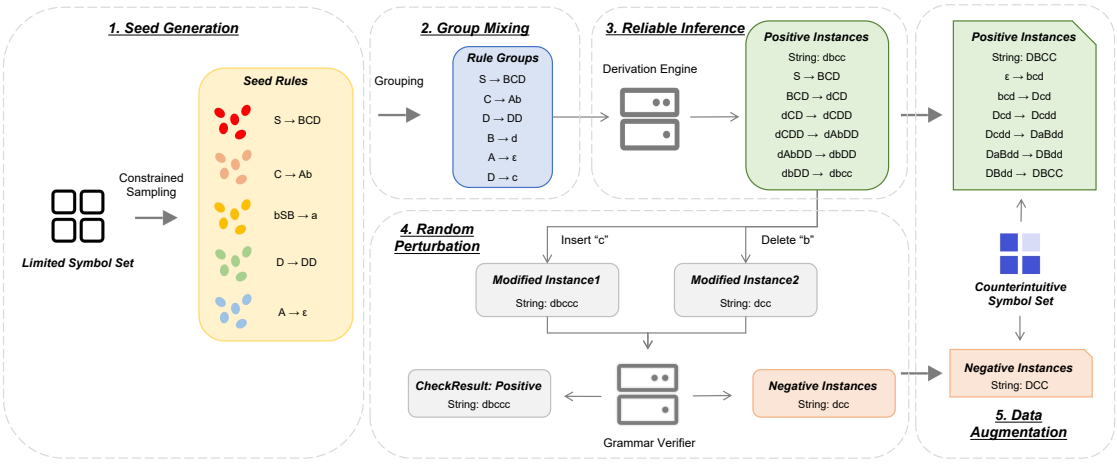

Figure 2: The data generation pipeline for FLoRE. Beginning with a limited symbol set, we progressively construct seed rules and rule groups, from which we derive both positive and negative sample data through reasoning.

Type	Initial	Null	Terminals	Nonterminals	Grammar Example	String Example
Normal	S	ε	Lowercase letters	Uppercase letters	S $\rightarrow$ SAE | ab E $\rightarrow$ Cb | CC | EE bA $\rightarrow$ a C $\rightarrow$ ε	aab
Swapped	ε	P	Uppercase letters	Lowercase letters	ε $\rightarrow$ ε ae | AB e $\rightarrow$ cB | cc | ee Ba $\rightarrow$ A c $\rightarrow$ P	AAB

Table 3: Examples of the two grammar forms, along with the strings they generate. "Normal" refers to the standard form of formal language grammar, whereas "Swap" denotes the transformed grammar obtained through the procedure described in Section 3.3.

of rule groups. To ensure the validity of derivations within each rule group, it is required that every group contains at least one Root rule. Subsequently, a number of rules are randomly sampled from the Context-Free and Context-Sensitive grammars. Finally, additional Recursive and Epsilon grammars are randomly selected and added to the group.

Reliable Inference. We develop a **Grammar Derivation Engine** that systematically applies production rules to generate multiple positive samples for each rule group. Each positive sample consists of a grammar rule set, a derived string, and its associated derivation process. We define a hyperparameter λ as the number of derivation steps, which is used to regulate the complexity of the generated data.

Random Perturbation. To obtain negative samples, we first apply perturbations to the positive samples by randomly inserting or deleting a number of characters. We then develop a **Grammar Verifier** employing a breadth-first search to test the perturbed positive samples. This verifier filters out any samples that, despite perturbation, still conform to the original grammar and thus remain valid. However, for some complex grammar types, the search space required to determine whether a string can be derived from the grammar is unbounded (Hopcroft et al., 2001). To address this issue, we define a hyperparameter μ as the upper bound of derivation steps.

Building upon the aforementioned data construction procedure, we subsequently divide the dataset according to specific task configurations. For formal language recognition, each data item consists of a grammar rule set, a string, the correct label, and the corresponding derivation process. If the label is false, the derivation process is left empty. For grammar induction, each data item consists of a set of input strings and a set of test strings, all of which are generated through multiple deriva-

tions from the same grammar rule set. The data examples for both tasks are presented in Figure 1.

Data Augmentation. To further disrupt the model's prior assumptions based on frequent exposure to standard grammatical conventions, encouraging genuine reasoning over memorized patterns, we additionally construct a batch of data for the formal language recognition task, and apply the following substitutions to the meanings of the symbols.

First, we replace the commonly used start symbol "S" with "ϵ". Next, we replace the symbol representing the empty string, "ϵ", with the letter "P". Furthermore, we reverse the standard convention by using uppercase letters to represent terminal symbols and lowercase letters for non-terminal symbols, deviating from the typical notation. Examples of the two grammatical forms are shown in Table 3.

3.4 Evaluation Metrics

For different tasks, we have designed several evaluation metrics accordingly.

Formal Language Recognition For FLR, we use the discrimination *accuracy (Acc)* as the evaluation metric. Let $D = (w_i, y_i)_{i=1}^{N}$ be a labeled dataset of strings, where each $w_i \in \Sigma$ is an input string and $y_i \in 0, 1$ is the ground-truth label indicating membership in the language L: $y_i = 1$ if $w_i \in L$, $y_i = 0$ if $w_i \notin L$.

Let $\hat{y}_i$ denote the predicted label output by the recognizer for w_i:

$$\hat{y}_i = \begin{cases} 1, & \text{if recognizer accepts } w_i \\ 0, & \text{otherwise} \end{cases}$$

Then, the *accuracy(Acc)* is defined as the fraction of correctly classified strings:

$$\text{Acc} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}(\hat{y}_i = y_i)$$

where $\mathbf{1}(\cdot)$ is the indicator function, equal to 1 if the condition is true and 0 otherwise.

Grammar Induction For GI, we use the *Pass* and *Generalization* as the evaluation metric for this task. Let G be the induced grammar, and let $S_{\text{in}} = w_1, w_2, \ldots, w_{N_{\text{in}}}$ be the input set of strings, $S_{\text{test}} = v_1, v_2, \ldots, v_{N_{\text{test}}}$ be the test set of strings. Define the derivability indicator function for a string x:

$$\mathbf{D}_G(x) = \begin{cases} 1, & \text{if } x \in L(G) \\ 0, & \text{otherwise} \end{cases}$$

where $L(G)$ is the language generated by grammar G. We define the proportion of input strings derivable by G as *Pass*. This metric quantifies how well G covers the input data, which is formulated as:

$$\text{Pass} = \frac{1}{N_{\text{in}}} \sum_{i=1}^{N_{\text{in}}} \mathbf{D}_G(w_i)$$

We define the proportion of test strings derivable by G as *Generalization*. This metric measures the ability of G to generalize beyond the input set to unseen data, which is formulated as:

$$\text{Generalization} = \frac{1}{N_{\text{test}}} \sum_{j=1}^{N_{\text{test}}} \mathbf{D}_G(v_j)$$

Process Assessment To comprehensively and accurately evaluate the reasoning process of LLMs, we introduce *Logical Consistency Rate (LCR)* as the evaluation metric, which is defined as the proportion of logically consistent inference processes: Let $\mathcal{P} = P_1, P_2, \ldots, P_N$ be a set of N reasoning processes. Each reasoning process P_i consists of a sequence of reasoning steps that aim to derive a target string $t_i \in \Sigma^*$ from a given starting point, using the given formal language grammar $\mathcal{G}$. Define the binary predicate:

$$\text{Consistent}(P_i) = \begin{cases} 1, & \text{if } P_i \text{ correctly applies} \\ & \text{rules in } \mathcal{G} \text{ and derives } t_i \\ 0, & \text{otherwise} \end{cases}$$

Then the *Logical Consistency Rate (LCR)* is defined as:

$$\text{LCR} = \frac{1}{N} \sum_{i=1}^{N} \text{Consistent}(P_i)$$

4 Experiments

4.1 Experimental Setups

We conduct a series of experiments on current mainstream LLMs using the constructed dataset. Our evaluation focuses on two primary objectives: (1) to assess whether the models can perform precise and rigorous deductive reasoning by correctly determining the membership relation between a given grammar and string; and (2) to examine their ability to carry out effective inductive reasoning by identifying underlying patterns and common rules from a set of example strings.

Acc / LLMs Type	Metric	Qwen2.5 7B	Qwen2.5 14B	LlaMa3 8B	Deepseek-R1 Distilled-Qwen-7B	Qwen plus	GPT 4o	Deepseek R1	Qwen3
Normal	ACC+	0.559	0.568	0.439	0.583	0.798	0.509	**0.806**	0.482
	ACC-	0.532	0.634	0.454	0.351	0.482	0.757	**0.843**	0.639
Swapped	ACC+	0.168	0.212	0.132	0.211	**0.635**	0.262	0.586	0.152
	ACC-	0.830	0.845	0.846	0.383	0.638	0.882	0.863	**0.936**

Table 4: Test result of FLR task. We evaluate the recognition accuracy of positive and negative samples on two different types of grammars.

Metric	Qwen 2.5-7B	LlaMa 3-8B	GPT 4o	Deepseek R1
Acc	0.626	0.796	0.595	0.729
LCR	0.214	0.053	0.929	**1.000**

Table 5: Test results for prompting the model to output both the classification result and reasoning path. Acc refers to the classification accuracy. LCR refers to the proportion of logically consistent inference processes on correct samples.

Our evaluation includes three types of LLMs: (1) open-source models without deep-reasoning capabilities, including Qwen2.5-7B-Instruct, Qwen2.5-14B-Instruct and LlaMa-3-8B; (2) open-source models designed for deep-reasoning, including Deepseek-r1 , Qwen3-8B and Deepseek-r1-distilled-Qwen2.5-7B-Instruct; (3) closed-source models without specialized reasoning training, including GPT-4o and Qwen-plus.

Additional implementation details, including the test-time prompts and model-specific evaluation settings, can be found in the Appendix C.

4.2 Main Results

Formal Language Recognition We first evaluate the performance of commonly used large language models on the formal language recognition task, with the results shown in Table 4. These LLMs are tested on two different grammar forms mentioned in Section 3.3, evaluating their accuracy in classifying both positive and negative sample datasets. Our findings are as follows.

1). Deep thinking models, such as Deepseek-R1, tend to perform better than non-inferential models on this task, which demonstrates their higher potential in complex abductive and deductive reasoning.

2). The logical reasoning capabilities of closed-source models in symbolic systems are not necessarily superior to those of small-scale open-source models.

3). The most intriguing finding is that, **when the grammatical form deviates from the common forms**, on one hand, all the tested models exhibit a significant performance decrease on positive samples, even when such changes in form are limited to symbol transformations. On the other hand, when the characters of the grammar change to an unfamiliar form, all the large models tend to classify the strings as non-inferable.

However, it remains unclear whether this performance discrepancy stems purely from limitations in reasoning ability, or from the model's inability to retain newly introduced character rules throughout the reasoning process. It is particularly difficult to determine because changing the character set requires the addition of explanatory text to define the function of the new characters(As is shown in Appendix C.2). This may need further research.

4). The local deep-thinking models appear to lack awareness of when to terminate their reasoning. Even though we have set a large output window, they often engage in excessive reasoning on negative samples, which results in incomplete classification outputs.

Grammar Induction We then evaluate the LLM's ability to identify general patterns from input strings and to induce plausible grammatical rules, the results are shown in Table 6. For each LLM, we input a set of strings derived from the same grammar, and require it to output a grammar capable of generating the input strings. Then, we apply the automated Grammar Verifier to evaluate the grammars produced by the models, calculating the pass rate of input strings and test strings for each test. Our results indicate that although there are still considerable performance differences among the reasoning-oriented models, these LLMs significantly outperform non-reasoning ones; however, even the best-performing reasoning models still fail to achieve strong generalization in their inductive results.

Performace	Qwen2.5 7B	Qwen2.5 14B	LlaMa3 8B	Deepseek-R1 Distilled-Qwen-7B	Deepseek R1	Qwen plus	GPT 4o	Qwen3
Pass	0.158	0.122	0.123	0.327	0.621	0.172	0.262	**0.859**
Generalization	0.037	0.032	0.020	0.189	0.144	0.028	0.036	**0.462**

Table 6: Test result of GI task. For each test case, we calculate the proportion of the input string that can be derived from the grammar summarized by the model as Pass, and that of test string is referred to as Generalization.The average value across all test cases is taken as the final performance of each model.

4.3 Evaluation of the Reasoning Process

In Section 4.2, we evaluate the model's performance on a classification task that requires logical reasoning. However, an important question is, under what conditions will the model produce a 'correct' judgment: is it genuinely **reasoning** through a valid path, or simply **guessing** based on the distribution of terminal character patterns?

To explore this question, we introduce additional constraints during inference to compel the model to explicitly generate its reasoning path. We perform a series of experiments, with the results shown in Table 5. Requiring the models to explicitly generate a reasoning path during its prediction process helps to improve their classification accuracy. This indicates that the revised prompts guide the model toward a more structured reasoning process, thereby promoting consistency in its reasoning.

We manually examine the reasoning paths extracted from the samples correctly classified by the models and find significant differences across model scales: Smaller models often **struggle** to produce coherent reasoning paths, even when their classification performance does not appear to be mere guessing. This may be due to the limited capacity of small models, which prevents them from backtracking and constructing a complete reasoning tree. Alternatively, they might simply be guessing answers based on the distribution of symbol sets and approximate pattern matching. The exact cause may require further investigation. In contrast, larger models tend to provide correct reasoning paths, despite occasional errors in notation or symbol confusion.

This indicates that model scale has a significant impact on the quality of reasoning, even when models with substantially different sizes may achieve similar performance in classification tasks. Case study examples for these attempts are provided in Appendix B.

Acc	Qwen 2.5-7B	LlaMa 3-8B	GPT 4o	Deepseek R1
Original	0.559	0.439	0.509	**0.806**
J & P	**0.626**	0.**796**	0.595	0.729
Guided	0.592	0.726	**0.590**	0.791

Table 7: Test results for prompting the model to follow human problem-solving steps. Original refers to the original prompt formulation, and Guided refers to the human-steps-following prompt formulation. J & P refers to the prompting format that elicits both the final judgment and the reasoning path.

4.4 Guided Prompting

When we prompt LLM to explicitly output their reasoning paths, their accuracy on judgment tasks improves to some extent. This raises an intriguing question: if we guide the model, through prompts, to solve problems by following the same step-by-step approach as humans, would its decision-making performance further improve? We conduct another experiment, with results shown in Table 7. For non-deep-reasoning models, prompting large models to engage in deeper reasoning can improve their performance on deductive reasoning tasks; however, for deep-reasoning models, such prompts may have the opposite effect.

5 Conclusions

Through careful experimentation, we find that existing mainstream large models fail to achieve satisfactory performance on logical reasoning tasks that are based on formal language grammars. Regardless of model type, performance on reasoning tasks consistently declines when the definitions of symbols in the rules diverge from their commonsense interpretations. Prompting can guide non-reflective models to generate reasoning processes resembling reflective thinking, thereby improving their performance.All of these findings provide a foundation for further improvements in the capabilities of LLMs.

Limitations

In this paper, we leverage formal language grammars to effectively address issues found in traditional logic reasoning evaluations, such as interference of commonsense knowledge. However, several limitations remain:

1). Insufficient evaluation of reasoning processes in formal language recognition. The most reasonable evaluation of logical reasoning should include both process evaluation and outcome evaluation. However, since process evaluation requires models to follow complex formatting instructions, it is currently difficult to automate and can only be performed through manual sampling and inspection. Therefore, we currently use classification accuracy as the evaluation metric. This limitation highlights the need for more systematic and scalable process-level evaluation methods in future work.

2). Limitations in the design of generalization testing for formal language induction. Although generating strings from the same grammar provides a relatively fair evaluation framework, expecting models to infer universal rules from a small number of examples is inherently difficult. This reflects a broader challenge shared across all inductive tasks — the inherent tension between data sparsity and generalization. Future work should explore more principled approaches to designing generalization benchmarks that better capture meaningful inductive reasoning.

References

Elham Aghakhani, Lu Wang, Karla T Washington, George Demiris, Jina Huh-Yoo, and Rezvaneh Rezapour. 2025. From conversation to automation: Leveraging large language models to analyze strategies in problem solving therapy. *arXiv preprint arXiv:2501.06101*.

V Aho Alfred, S Lam Monica, and D Ullman Jeffrey. 2007. *Compilers principles, techniques & tools*. pearson Education.

Prajjwal Bhargava and Vincent Ng. 2022. Commonsense knowledge reasoning and generation with pretrained language models: A survey. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 12317–12325.

Michael K Chen, Xikun Zhang, and Dacheng Tao. 2025. Justlogic: A comprehensive benchmark for evaluating deductive reasoning in large language models. *arXiv preprint arXiv:2501.14851*.

Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Johan Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, and 1 others. 1996. Using the framework. Technical report, Technical Report LRE 62-051 D-16, The FraCaS Consortium.

Armin Cremers and Seymour Ginsburg. 1975. Context-free grammar forms. *Journal of Computer and System Sciences*, 11(1):86–117.

Colin De la Higuera. 2010. *Grammatical inference: learning automata and grammars*. Cambridge University Press.

Shujie Deng, Honghua Dong, and Xujie Si. 2024. Enhancing and evaluating logical reasoning abilities of large language models. In *ICLR 2024 Workshop on Secure and Trustworthy Large Language Models*.

Gabriel Ebner. 2021. *Inductive theorem proving based on tree grammars*. Ph.D. thesis, Technische Universität Wien.

J St BT Evans, Julie L Barston, and Paul Pollard. 1983. On the conflict between logic and belief in syllogistic reasoning. *Memory & cognition*, 11(3):295–306.

Jerry A Fodor and Zenon W Pylyshyn. 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71.

E Mark Gold. 1967. Language identification in the limit. *Information and control*, 10(5):447–474.

Morgan Gray, Jaromir Savelka, Wesley Oliver, and Kevin Ashley. 2024. Using llms to discover legal factors. In *Legal Knowledge and Information Systems*, pages 60–71. IOS Press.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, and 1 others. 2022. Folio: Natural language reasoning with first-order logic. *arXiv preprint arXiv:2209.00840*.

Michael A Harrison. 1978. *Introduction to formal language theory*. Addison-Wesley Longman Publishing Co., Inc.

John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. 2001. Introduction to automata theory, languages, and computation. *Acm Sigact News*, 32(1):60–65.

Wenyue Hua, Kaijie Zhu, Lingyao Li, Lizhou Fan, Shuhang Lin, Mingyu Jin, Haochen Xue, Zelong Li, JinDong Wang, and Yongfeng Zhang. 2024. Disentangling logic: The role of context in large language model reasoning capabilities. *arXiv preprint arXiv:2406.02787*.

Tim Hunter. 2021. The chomsky hierarchy. *A companion to Chomsky*, pages 74–95.

Tao Jiang, Ming Li, Bala Ravikumar, and Kenneth W Regan. 2009. Formal grammars and languages. In *Algorithms and Theory of Computation Handbook, Volume 1*, pages 549–574. Chapman and Hall/CRC.

Hanmeng Liu, Zhizhang Fu, Mengru Ding, Ruoxi Ning, Chaoli Zhang, Xiaozhang Liu, and Yue Zhang. 2025. Logical reasoning in large language models: A survey. *arXiv preprint arXiv:2502.09100*.

Hanmeng Liu, Jian Liu, Leyang Cui, Zhiyang Teng, Nan Duan, Ming Zhou, and Yue Zhang. 2023. Logiqa 2.0—an improved dataset for logical reasoning in natural language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:2947–2962.

Mohamed Nejjar, Luca Zacharias, Fabian Stiehle, and Ingo Weber. 2025. Llms for science: Usage for code generation and data analysis. *Journal of Software: Evolution and Process*, 37(1):e2723.

Mihir Parmar, Nisarg Patel, Neeraj Varshney, Mutsumi Nakamura, Man Luo, Santosh Mashetty, Arindam Mitra, and Chitta Baral. 2024. Logicbench: Towards systematic evaluation of logical reasoning ability of large language models. *arXiv preprint arXiv:2404.15522*.

Nisarg Patel, Mohith Kulkarni, Mihir Parmar, Aashna Budhiraja, Mutsumi Nakamura, Neeraj Varshney, and Chitta Baral. 2024. Multi-logieval: Towards evaluating multi-step logical reasoning ability of large language models. *arXiv preprint arXiv:2406.17169*.

Josef Pecht. 1983. On the real-time recognition of formal languages in cellular automata. *Acta Cybernetica*, 6(1):33–53.

Frank Pfenning. 2004. Automated theorem proving. *Lecture notes, March*.

Abulhair Saparov, Richard Yuanzhe Pang, Vishakh Padmakumar, Nitish Joshi, Mehran Kazemi, Najoung Kim, and He He. 2023. Testing the general deductive reasoning capacity of large language models using ood examples. *Advances in Neural Information Processing Systems*, 36:3083–3105.

CH Sumitha and Krupa Ophelia Geddam. 2011. Implementation of recursively enumerable languages in universal turing machine. *International Journal of Computer Theory and Engineering*, 3(1):153.

Alon Talmor, Oyvind Tafjord, Peter Clark, Yoav Goldberg, and Jonathan Berant. 2020. Leap-of-thought: Teaching pre-trained models to systematically reason over implicit knowledge. *Advances in Neural Information Processing Systems*, 33:20227–20237.

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 and Data Engineering*.

Zonglin Yang, Li Dong, Xinya Du, Hao Cheng, Erik Cambria, Xiaodong Liu, Jianfeng Gao, and Furu Wei. 2022. Language models as inductive reasoners. *arXiv preprint arXiv:2212.10923*.

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 preprint arXiv:2104.06598*.

Qin Zhu, Fei Huang, Runyu Peng, Keming Lu, Bowen Yu, Qinyuan Cheng, Xipeng Qiu, Xuanjing Huang, and Junyang Lin. 2025. Autologi: Automated generation of logic puzzles for evaluating reasoning abilities of large language models. *arXiv preprint arXiv:2502.16906*.

A NLI vs Logical Reasoning

Natural language inference (NLI) and logical reasoning tasks differ fundamentally in the nature of the knowledge they require. NLI tasks involve determining the relationship between a premise and a hypothesis—typically labeled as entailment, contradiction, or neutrality—based on both linguistic cues and extensive background or commonsense knowledge. For instance, correctly inferring that "The man is running" entails "The man is moving" often relies on an implicit understanding of physical actions and their typical consequences, rooted in human experience.

In contrast, logical reasoning tasks aim to abstract away from such language-specific or world knowledge, focusing instead on the formal structure of reasoning. These tasks are often designed to minimize reliance on lexical or contextual cues and instead emphasize the manipulation of well-defined logical relations, such as conjunction, disjunction, and implication. As a result, logical reasoning tasks require models to exhibit systematic, rule-based inference capabilities rather than drawing on broad linguistic commonsense. Therefore, to ensure validity and interpretability, logical reasoning tasks must reduce the dependence on natural language knowledge and focus on explicit, domain-contained rules and symbols.

However, existing evaluation methods for logical reasoning unavoidably introduce some degree of commonsense knowledge during data construction. For example, we illustrate this with a sample from the LogiQA dataset. Among the four options below, Option 2 and Option 4 are clearly inconsistent with commonsense knowledge. Therefore, LLM can eliminate these two choices based on its stored knowledge rather than reasoning ability. On

the other hand, if a LLM already knows from common knowledge that "Cantonese are southerners", it might accept the argument as obviously valid without consciously identifying the logical structure. This makes it hard to tell whether it is using deductive reasoning or just relying on background knowledge.

context:
Some Cantonese don't like chili, so some southerners don't like chili.

query:
Which of the following can guarantee the above argument?

options:
1.Some Cantonese love chili.
2.Some people who like peppers are southerners.
3.All Cantonese are southerners.
4.Some Cantonese like neither peppers nor sweets.

B Case Study Results

It remains challenging to obtain accurate analysis results through automatic parsing models' reasoning path. As a formatted reasoning path imposes relatively complex constraints, and any single error can cause the entire reasoning process to fail.

Our observations suggest that the instruction-following capabilities of non-deep-thinking LLMs' are often insufficient to reliably accomplish such tasks. Consequently, when applying rules based on a strict symbolic system for matching, the output paths of these models are often prone to error. Therefore, we conduct a series of case studies as mentioned in the context. We sample and manually check two kinds of results:

1. Asking the LLMs to parse only the reasoning path. 2.Asking the LLMs to both judge the attribution and parse the reasoning path simultaneously. Examples and conclusions are as follows.

B.1 Parse only the reasoning parse

Our first attempt is to prompt the large language model to only parse the reasoning path from the input string and grammar; However, when we ask the LLMs to do so, many LLMs often resorted to heuristic shortcuts—fabricating rules or generating superficially plausible reasoning paths, even for negative samples they would have otherwise classified correctly. As is shown below, the Qwen2.5-7B model fabricated the rule of $A \rightarrow ada$.

rules:
$S \rightarrow EBS|aD$,
$E \rightarrow C|Aes|eBc$,
$B \rightarrow bcd|ada|\epsilon|BB$,
$dSA \rightarrow E$,
$D \rightarrow \epsilon$,
$A \rightarrow \epsilon$,

string:
esadaa

path:
1. S
2. EBS ($S \rightarrow EBS$)
3. AesB ($E \rightarrow Aes$)
4. adaB ($\underline{A \rightarrow ada}$)
5. adabc ($B \rightarrow bcd$)
6. esadaa ($D \rightarrow \epsilon$, $B \rightarrow \epsilon$)

B.2 Recognition and Parse

Meanwhile, when we ask the LLMs to both conduct recognition and parse the reasoning path simultaneously, smaller LLMs struggle to output reasonable reasoning paths, even when their judgments are correct. There are several common types of mistake.

(1) The first type is symbol confusion. As shown below, the model fail to distinguish between terminal symbols and non-terminal symbols during output. If we disregard such confounding factors and attempt to apply the correct rules to interpret the reasoning path of large models, we find that when inferring the string **bcdeBCd**, it considered the result to be correct, but the actual target string is **bcdebcd**.

rules:
$S \rightarrow BEd$,
$B \rightarrow bcd|ada$,
$E \rightarrow b|CB|Aes|eBc|\epsilon|EE$,
$bA \rightarrow A$,
$A \rightarrow \epsilon$

path:
1.(s, s→bed, 0) BEd
2.(bed, b→bcd, 0) bcdEd
3.(bcdd, e→ebc, 1) <u>bcdeBcd</u>

(2)The second type is rule fabrication, where the model randomly invents non-existent rules and applies them to a certain step of the substitution process. As is shown below, the LLM fabricates the rule *c* -> ϵ and *bsb* -> *ada*, and symbol confusion exists meanwhile.

rules:

$S \rightarrow EBS|Ca|aD$
$E \rightarrow C|CB|Aes|bE$
$bSb \rightarrow Ada"$
$D \rightarrow \epsilon$
$D \rightarrow \epsilon$
$B \rightarrow \epsilon|BB$
$A \rightarrow \epsilon"$

path:

(cbs, e → c, 1)
(cs, b → ϵ, 2)
(s, c → ϵ, 1)
(ebs, s → ebs, 0)
(aes, e → aes, 1)
(es, a → ϵ, 1)
(sebe, e → be, 1)
(es, bsb → ada, 0)

(3) The third type is rule application errors, where the large model applies correct rules to incorrect symbols during substitution. As is shown below, in step 5, LlaMa3-8B applies the rule "d -> ϵ" to position 0, while the correct application should be at position 2.

rules:

1. (s, s → bed, 0)
2. (bed, e → ϵ, 1)
3. (bd, b → ada, 0)
4. (adad, d → ϵ, 1)
5. (aad, d → ϵ, 0)
6. (ad, d → ϵ, 0)

For larger LLM, the proportion of correctly generated reasoning paths is significantly higher than that of smaller models, suggesting that they are indeed capable of performing step-by-step reasoning based on rule-like systems to a certain extent. As shown below, despite occasional slips (such as writing **"acsSSE**" as **"acSSE"**), Deepseek-R1 demonstrates strong consistency in reasoning on samples that are classified correctly. Moreover, this symbolic error occurs only once and is immediately corrected in the following step, suggesting that it resembles a slip of the pen by a human student—a minor writing mistake that does not compromise the overall reasoning process.

path:

('S', 'S->SAE', 0)
('SAE', 'S->Cac', 0)
('CacAE', 'C->ϵ', 0)
('acAE', 'A->sS', 2)
('acsSE', 'S->SS', 3)
('acSSE', 'S->Cac', 3)
('acsCacSE', 'C->ϵ', 3)
('acsacSE', 'S->Cac', 5)
('acsacCacE', 'C->ϵ', 5)
('acsacacE', 'E->b', 7)

B.3 Preliminary testing on commercial LLMs

To investigate the types of grammar-related logical reasoning tasks where current large models are prone to errors, we manually collected several formal-language-related problems and conducted sampling tests on some commercial large models. The specific details are provided in the appendix.

We find that, currently, commercial LLMs can correctly handle certain tasks involving simple, short-path reasoning on context-free grammars(CFG)(Cremers and Ginsburg, 1975). Common errors are concentrated in two types of grammars: (1) Left-recursive grammars, where a nonterminal symbol can derive an expression that contains itself through a specific production rule, leading to infinite recursion during parsing. (2) Context-sensitive grammars, which often involve production rules with character reduction steps.

C Detailed Experimental Setups

Due to limited resources, we conduct data sampling for experiments across different tasks and model types, taking into account factors such as inference speed, GPU memory capacity, and cost.

To ensure the reliability of the reasoning process, all locally deployed models were executed via the Transformers library, with the maximum new token length set to 8192 tokens, which should be considered adequate for the expected inference depth. All results are the average values of the model's performance on the test set.

C.1 Hyperparameters

1. In data-construction, to balance task complexity and computational feasibility, we constrain λ for each data instance to the range of 8 to 10. This range is chosen to ensure that the reasoning process is non-trivial—requiring more than simple pattern matching—while still remaining within the

manageable token limits of current large language models. Derivations of this length are sufficiently deep to reveal recursive or context-dependent structures, enabling meaningful evaluation of deductive reasoning capabilities. At the same time, the length is moderate enough to avoid unnecessary computational overhead and to ensure that all derivation traces can be fully processed during both generation and verification. Additionally, longer derivation paths yield more structurally diverse samples, which enhances the quality of both positive and perturbed negative instances, allowing for a more robust assessment of the models' generalization and discrimination abilities. 2. Accordingly, we set ϵ to 10. This operation transforms the context-related grammar search space into a finite one, thereby enabling the identification task to be decided by a rule-based parser.

C.2 Prompts

We design a set of task-specific prompts for our experiments. Here we provide a detailed presentation of these prompts. For each example, we present a comparison between Chinese and English, with all experiments conducted using the Chinese version. All experiments are done with a zero-shot setting.

Normal type FLR(ZH)

rules: rules string: string task: 从S出发，根据规则，能否推理出目标字符串？返回“Yes”或“No”

Normal type FLR(EN)

rules: rules string: string task: Starting from S, can the target string be derived according to the rules? Return "Yes" or "No".

Swapped type FLR(ZH)

给定一组语言文法规则如下： rules 目标字符串为：string 符号说明：ε 表示推导的起点 P表示空字符（即一个替代符号，用来表示空）。 请回答，是否可以由该文法推导出来目标字符串。请分析推导过程，必要时列出每一步，并在最后给出你的结论：yes（如果可以推导出）结论：no（如果不能推导出）

Swapped type FLR(EN)

Given a set of grammar rules as follows: rules The target string is: string Notation: ε denotes the starting symbol of the derivation. P represents the empty string (i.e., a placeholder symbol indicating emptiness). Please determine whether the target string can be derived from the grammar. Analyze the derivation process, listing each step if necessary, and provide your final conclusion: Conclusion: yes (if the target string can be derived) Conclusion: no (if the target string cannot be derived)

Parse Only the Reasoning Path(ZH)

你将得到一个形式语言文法和一个目标字符串。请推导该字符串是如何由该文法生成的，展示清晰的推导步骤。每一步请注明使用了哪一条产生式规则（例如：S → aSb）。
文法： rules
字符串： string
只输出推导过程，包括每一步使用的产生式规则。
格式如下： 1. S（初始符号）

Parse Only the Reasoning Path(EN)

You will be given a formal grammar and a target string. Please derive how the string can be generated from the grammar, and show the derivation steps clearly. For each step, indicate the production rule used (e.g., S → aSb).
Grammar: rules
String: string
Only output the derivation process, including the production rule used in each step.
Format:
1. S (start symbol)

Normal type GI(ZH)

请根据以下输入的字符串集合，推导并生成一个符合形式语言规范的上下文无关文法。请严格按照以下要求执行： 1. 文法推导方法： - 分析字符串的最小公共模式 - 识别递归结构和重复模式 - 提取终结符和非终结符 - 合并相似的产生式规则
2. 输出格式要求： - 每行一个产生式规则 - 使用->符号连接左右部 - 非终结符用大写字母表示 - 终结符用引号包裹或小写字母表示 - 避免任何额外解释文本
输入字符串： input strings
请严格按照上述格式要求，只返回推导出的文法规则，每个规则独立一行，不使用编号和多余符号。

Normal type FLR With Path (ZH)

给定一组形式语言文法规则和一个目标字符串，判断目标字符串是否可以从起始符号 start symbol 推导得到。如果可以，返回 'yes' 并给出从start symbol开始的推导步骤，每次替换的格式如下：

(当前串, 使用的规则, 替换位置)
其中，“当前串”指的是此次替换前的字符状态，使用的规则”指此次替换使用的文法规则，“替换位置”指的是该规则应用于当前串中的第几个符号位置（从0开始）；

如果不能推导出该串，返回 "no"。

文法规则如下（每行一条）：
grammar

目标字符串：string

Normal type GI(EN)

Please derive and generate a context-free grammar (CFG) that conforms to formal language conventions based on the following set of input strings. Follow the instructions below strictly:
1. Grammar Derivation Method:
Analyze the minimal common patterns among the strings
Identify recursive structures and repeating patterns
Extract terminals and non-terminals
Merge similar production rules
2. Output Format Requirements:
One production rule per line
Use -> to connect the left-hand side and the right-hand side
Represent non-terminals with uppercase letters
Represent terminals using quotes or lowercase letters
Avoid any additional explanatory text
Input strings: input strings
Please strictly follow the format above and return only the derived grammar rules. Each rule should appear on a separate line, with no numbering or extra symbols.

Normal type FLR With Path (EN)

Given a set of formal grammar rules and a target string, determine whether the target string can be derived from the start symbol start symbol.
If it can be derived, return "yes" and provide the derivation steps starting from the start symbol. Each replacement step should follow the format:
(current string, applied rule, replacement position) Where:
"current string" refers to the state of the string before the replacement,
"applied rule" refers to the grammar rule used in this step,
"replacement position" refers to the index (starting from 0) of the symbol in the current string where the rule is applied.
If the target string cannot be derived, return "no".
Grammar rules (one per line): grammar
Target string: string

Normal type FLR With Guided Prompting(ZH)

给定一个形式文法和一个目标字符串，请按照以下步骤判断该字符串是否可以由该文法生成：
列出文法的产生式规则，明确起始符号。
分析文法结构，判断是否递归、包含空产生式、是否对称等。
观察目标字符串的长度和结构，尝试寻找与文法规则的匹配模式。 选择推导方向：
使用 自顶向下推导（从起始符号出发）逐步替换非终结符，尝试构造目标字符串；
或 自底向上归约（从目标字符串出发）寻找文法右部，逐步归约回起始符号。
记录推导过程（或构建推导树），确保每一步都合法。
得出结论：
如果成功推导出完整的目标字符串，则说明该字符串属于文法的语言,返回“Yes”。
如果无法推导或中途卡住，则不属于，返回“No”。
文法：rules 目标字符串：string"

Normal type FLR With Guided Prompting(EN)

Given a formal grammar and a target string, please determine whether the string can be generated by the grammar by following these steps:
1. List the production rules of the grammar and identify the start symbol.
2. Analyze the structure of the grammar: determine whether it is recursive, includes empty productions, or exhibits symmetry.
3. Examine the length and structure of the target string, and try to identify patterns that match the grammar rules.
4. Choose a derivation strategy:
Use top-down derivation: start from the start symbol and iteratively replace nonterminals to construct the target string; Or use bottom-up reduction: start from the target string, find right-hand sides of the grammar, and iteratively reduce to the start symbol.
5. Record the derivation steps (or construct a derivation tree), ensuring that each step is valid according to the grammar.
6. Draw a conclusion:
If the target string can be completely derived, it belongs to the language defined by the grammar — return "Yes". If derivation fails or gets stuck, the string does not belong to the language — return "No".
Grammar: rules
Target string: string