LogicTree: Structured Proof Exploration for Coherent and Rigorous Logical Reasoning with Large Language Models

Anonymous ACL submission

Abstract

Large language models (LLMs) have achieved remarkable multi-step reasoning capabilities across various domains. However, LLMs still face distinct challenges in complex logical reasoning, as (1) proof-finding requires systematic exploration and the maintenance of logical coherence and (2) searching the right combination of premises at each reasoning step is inherently challenging in tasks with large premise space. To address this, we propose LogicTree, an inference-time modular framework employing algorithm-guided search to automate structured proof exploration and ensure logical coherence. Advancing beyond tree-of-thought (ToT), we incorporate caching mechanism into LogicTree to enable effective utilization of historical knowledge, preventing reasoning stagnation and minimizing redundancy. Furthermore, we address the combinatorial complexity of premise search by decomposing it into a linear process. The refined premise selection restricts subsequent inference to at most one derivation per step, enhancing reasoning granularity and enforcing strict step-by-step reasoning. Additionally, we introduce two LLM-free heuristics for premise prioritization, enabling strategic proof search. Experimental results on five datasets demonstrate that LogicTree optimally scales inference-time computation to achieve higher proof accuracy, surpassing chain-of-thought (CoT) and ToT with average gains of 23.6% and 12.5%, respectively, on GPT-4o. Moreover, within LogicTree, GPT-4o outperforms o3-mini by 7.6% on average.

1 Introduction

Recent advances in large language models (LLMs), such as OpenAI's o1/o3 series (OpenAI, 2024a, 2025), DeepSeek-R1 (Guo et al., 2025) and Grok-3 (xAI, 2025), have demonstrated remarkable reasoning capabilities in domains like code generation and complex mathematical problem-solving. However, *logical reasoning* (Dowden, 2020; Clark et al., 2020) presents unique challenges that differentiate it from other reasoning domains (Liu et al., 2025; Xu et al., 2025). It demands rigorous verification of a hypothesis through deliberate reasoning over a set of premises consisting of facts and rules, where two difficulties may arise. First, in complex problems, the precise proof path is not immediately apparent. Proof discovery requires systematic and extensive exploration (Saparov and He, 2023). Second, each reasoning step involves selecting relevant premises and inferring based on them. In a large premise space, difficulty in identifying the right fact-rule combination directly affects inference accuracy (Kazemi et al., 2023).

To tackle these challenges, some studies use an iterative framework to build longer reasoning chains for solving complex problems (Creswell et al., 2023). Within the framework, they adopt a modular approach to decompose individual reasoning steps (Khot et al., 2023), separating premise selection from inference and assigning each to specialized LLM modules for improved accuracy (Xu et al., 2024b; Zhang et al., 2024; Sun et al., 2024). Further research integrates LLM modules into tree structures, enabling systematic proof exploration (Kazemi et al., 2023; Yao et al., 2023; Wang et al., 2025). Although these methods have achieved notable advancements, there are still limitations:
(1) Difficulties in maintaining logical coherence and in effectively utilizing derived knowledge obstruct progressive proof construction. In some iterative approaches (Creswell et al., 2023; Zhang et al., 2024), reasoning steps are not required to build directly on prior derivations, which may disrupt logical coherence, hinder deep reasoning and cause redundancy (Wang et al., 2025). While tree-based method (Yao et al., 2023) mitigates this issue, it lacks mechanisms to share derived knowledge across branches, potentially leading to reasoning stagnation (Sun et al., 2024).
(2) Combinatorial complexity hinders precise

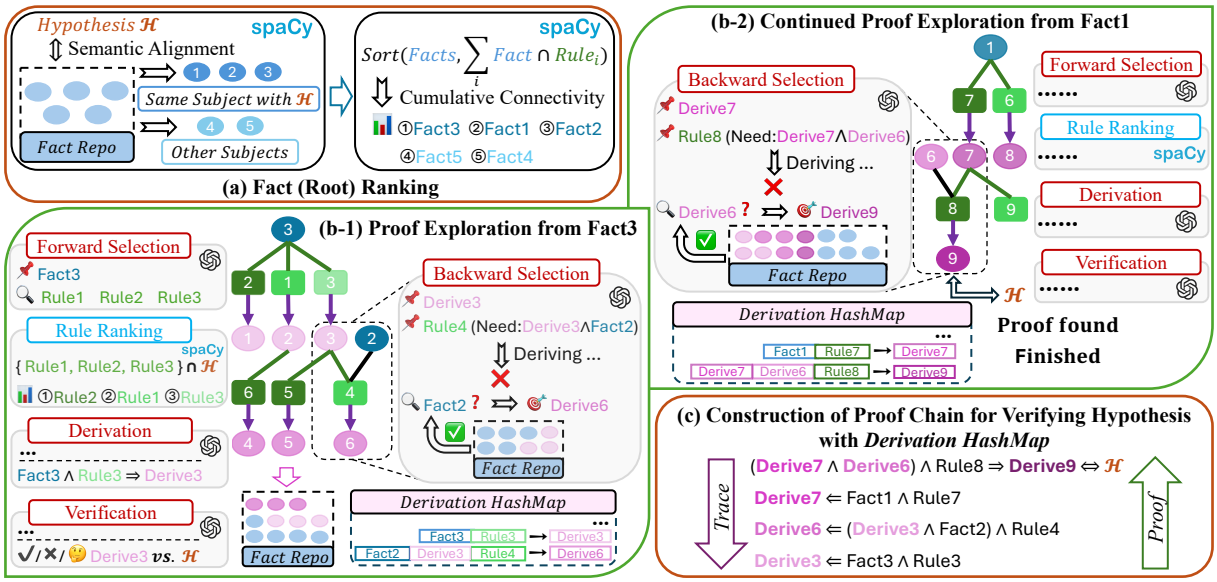


Figure 1: The overview of LogicTree: (a) Fact (root) ranking; (b) Tree-by-tree search for proof exploration: (b-1) Proof exploration from the top-ranked fact (Fact3) which has the highest cumulative connectivity with rules, (b-2) Continued proof exploration from the next ranked fact (Fact1); (c) Construction of proof chain. The framework consists of (i) two caches: Fact Repository (Fact Repo) and Derivation HashMap; (ii) four LLM-based modules: Forward Selection, Backward Selection, Derivation, Verification. Additionally, we leverage spaCy for fact and rule ranking. Within a tree: a blue oval represents a given fact; a green rectangle represents a rule; a purple oval represents a derived fact. LogicTree on an example from ProofWriter (Tafjord et al., 2020) is shown in Figure 5.

premise selection which is essential for accurate stepwise reasoning. At each step, the model must identify the right combination of facts and rules from premise space for subsequent inference. The combinatorial search increases the risk of imprecise selection, which results in failed or inaccurate inference (Kazemi et al., 2023; Liu et al., 2024).

(3) Employing LLMs for proof planning (Wang et al., 2023a) may be ineffective for complex logical reasoning, as such tasks require extensive and adaptive exploration. This often renders LLMs' planning unreliable and uninterpretable (Saparov and He, 2023; Kambhampati et al., 2024). Meanwhile, low-computation LLM-free heuristics may be sufficiently effective for strategic proof search, yet they remain largely overlooked.

To address these limitations, we propose LogicTree, a novel inference-time modular framework for structured proof exploration. The overview of our framework is shown in Figure 1. LogicTree includes four LLM-based modules: *Forward Selection*, *Backward Selection*, *Derivation* and *Verification*, which are embedded in tree structure. Additionally, we incorporate Fact Repository and Derivation HashMap into LogicTree as cache components. Fact Repository is initialized with given facts and dynamically stores derived facts. It enables branches to access the given facts and the derivations from earlier branches, facilitating cross-branch information flow and effective utilization of historical knowledge throughout proof exploration. Derivation HashMap records derived facts along with their derivation paths for a traceable reasoning process. We employ depth-first search (DFS) to orchestrate LLM modules and cache components, automating systematic proof search while ensuring logical coherence.

Furthermore, at each reasoning step, our framework decomposes the search for fact-rule combinations into *Forward (rule) Selection* followed by *Backward (fact) Selection*, reducing the complexity from combinatorial to linear. With this optimization, each selected rule-fact combination includes exactly one rule and its relevant fact(s), restricting inference to at most one derivation per step. This key improvement enhances reasoning granularity and enforces strict step-by-step reasoning, contributing to strengthened reasoning rigor.

Additionally, we introduce two heuristics leveraging spaCy[1] for premise prioritization: (1) *Fact (root) ranking* for global ordering of tree search; (2) *Rule ranking* at local level for early stopping in DFS. These LLM-free heuristics provide com-

[1]An open source NLP library (https://spacy.io/).

putationally efficient and interpretable strategies to accelerate proof-finding, avoiding blind and exhaustive search.

Our framework enables extensive exploration and fine reasoning granularity, optimally scaling reasoning length and inference-time computation (Snell et al., 2024) to enhance logical reasoning capability. Our evaluation on five challenging logical reasoning benchmarks demonstrates that LogicTree significantly outperforms chain-of-thought (CoT) (Wei et al., 2022) and other modular methods in proof accuracy. Furthermore, within LogicTree framework, both Llama-3.3 70B (Dubey et al., 2024) and GPT-4o (Achiam et al., 2023) surpass OpenAI's o1-mini and o3-mini models. In-depth analysis reveals that our approach facilitates precise premise selection and accurate inference at each reasoning step while minimizing redundancy.

The main contributions of our work are:

- We propose LogicTree, a novel inference-time framework that enables structured proof exploration while ensuring logical coherence. Additionally, we integrate cache components to effectively utilize historical knowledge and facilitate traceable reasoning process.
- We address combinatorial complexity in premise search and enhance reasoning granularity, improving stepwise reasoning accuracy and strengthening overall reasoning rigor.
- We introduce two LLM-free heuristics for premise prioritization, providing low-computation and interpretable strategies to improve proof-finding efficiency.

2 Related Work

2.1 Reasoning through Strategic Prompting

Pre-trained language models (Brown et al., 2020; Chowdhery et al., 2023; Touvron et al., 2023) exhibit emergent reasoning abilities with increasing model scale. Strategic prompt engineering techniques, such as CoT (Wei et al., 2022; Kojima et al., 2022), Auto-CoT (Zhang et al., 2023), self-consistency (Wang et al., 2023b), least-to-most (Zhou et al., 2023), help guide LLMs through intermediate reasoning steps, significantly improving LLM reasoning performance. However, the inherent simplicity of CoT and its variants, which is typically characterized by a left-to-right reasoning process with limited reasoning length, restricts their effectiveness in logical reasoning tasks that require exploration (Yao et al., 2023; Xie et al., 2023).

2.2 Inference-time Scaling for Reasoning

Just as human may take more time to carefully analyze a complex question, enabling LLMs to refine their response with deliberate reasoning and increased inference-time computation is crucial for developing intelligent reasoning systems (Snell et al., 2024; OpenAI, 2024a; Chen et al., 2025).

Reasoning models trained via reinforcement learning (RL). Applying large-scale RL in LLM post-training phase has proven highly effective in enhancing reasoning abilities. It enables LLMs to develop reflection, self-correction, and long-chain reasoning skills for problem-solving (OpenAI, 2024a; Kumar et al., 2024; Shao et al., 2024; Yeo et al., 2025). Recently, DeepSeek-R1 (Guo et al., 2025) made significant breakthrough by achieving strong reasoning performance purely through RL, without the need for supervised fine-tuning (SFT).

Modular inference without LLM parameter updates. Modular approach decomposes complex reasoning tasks into simpler sub-tasks, each assigned to specialized LLM modules implemented through few-shot prompting (Khot et al., 2023). In logical reasoning, it involves two key modules that operate iteratively: premise selection and inference (Creswell et al., 2023). Extending from this foundation, Cumulative Reasoning (Zhang et al., 2024) integrates LLM verifier to validate reasoning steps. DetermLR (Sun et al., 2024) employs LLM scorer to prioritize relevant premises. SymbCoT (Xu et al., 2024b) and Aristotle (Xu et al., 2024a) introduce LLM translator to convert natural language input into symbolic representations. Further research embeds LLM modules into topological structures, enabling deliberate problem solving (Yao et al., 2023; Besta et al., 2024).

Current LLM-based methods for logical reasoning still struggle to perform structured exploration while ensuring logical coherence and rigor in complex, multi-step reasoning tasks. To address these challenges, we propose a novel inference-time modular approach that enables systematic and extensive proof exploration and enhances reasoning rigor.

3 LogicTree for Logical Reasoning

3.1 Task Definition

Logical reasoning aims to determine the truth value (*true*, *false*, or *unknown*) of a *hypothesis* $\mathcal{H}$ based on a set of premises consisting of *facts* $\mathcal{F}$ and *rules* $\mathcal{R}$ (Dowden, 2020). An example is shown in Figure 5. Formally, $\mathcal{F} = \{f_i \mid i = 1, 2, \ldots, N_{\mathcal{F}}\}$,

where each f_i represents a definitive statement within the reasoning system. $\mathcal{R} = \{r_i \mid i = 1, 2, \ldots, N_{\mathcal{R}}\}$, where each r_i represents a conditional statement that defines a logical relationship between facts and inferred conclusions. The reasoning process applies standard logical operators, including: Negation ($\neg$), Conjunction ($\wedge$), Disjunction ($\vee$), Implication ($\Rightarrow$), Equivalence ($\Leftrightarrow$). We define the set of intermediate derived facts as $\mathcal{D} = \{d_i \mid i = 1, 2, \ldots, N_{\mathcal{D}}\}$.

3.2 Components of LogicTree Framework

As shown in Figure 1, the LogicTree framework includes (1) two caches: **Fact Repository** and **Derivation HashMap**; (2) four LLM-based modules: **Forward Selection**, **Derivation**, **Backward Selection**, and **Verification**, each implemented by few-shot prompting. The specific prompts for each module, along with example inputs and outputs, are provided in Appendix I.

Fact Repository and Derivation HashMap. Fact Repository is initialized with given facts $\mathcal{F}$ and continuously stores derived facts $\mathcal{D}$. It enables tree branches to access the given facts and earlier derivations. This facilitates cross-branch information flow and effective utilization of historical knowledge throughout proof exploration. Additionally, it checks whether a newly derived fact from a tree branch is unique among those already stored. If not, the branch is marked as a dead end to avoid redundancy and circular reasoning. Derivation HashMap stores derived facts as keys and their derivation paths as values, enabling a traceable reasoning process. Upon proof completion, the proof chain is reconstructed bottom-up, starting from the final path that verifies the hypothesis. If a fact in the path is found in the HashMap (i.e., it is derived rather than given), its associated derivation path is retrieved. This process occurs iteratively, constructing a streamlined proof as shown in Figure 1c.

Forward Selection Module. Based on a fact (either f_i or d_i), this module selects all the relevant **rules** from the given rule set $\mathcal{R}$. A rule is considered relevant if its condition(s) are fully or partially satisfied by the fact. Each selected rule is added as a child node of the fact, forming parallel branches in the tree structure.

Derivation Module. Along each branch, this module performs a strict one-step derivation using the current leaf rule and its parent fact. A successful derivation occurs if the fact fully satisfies the rule's condition. If the derivation fails, it results from one of the two reasons: (1) the fact does not satisfy the rule's condition at all, i.e., the rule was incorrectly selected by Forward Selection Module; (2) the fact partially satisfies the rule's conditions, with some required fact(s) still missing. In the first case, the branch is marked as a dead end. In the second case, where conjunctive reasoning (e.g., $f_1 \wedge f_2 \wedge r_1 \Rightarrow d_1$) is required, the branch is marked as a *pseudo* dead-end, where the missing fact(s) may still be retrievable.

Backward Selection Module. If a branch is marked as a pseudo dead-end, this module is queried to attempt rule completion and resolve the stagnation. This module uses the current fact-rule pair as a pivot to identify the missing **fact(s)** required for derivation. It then searches Fact Repository to determine their availability. The missing fact(s) may be a given fact f_i (Figure 1b-1) or a derived fact d_i from an earlier branch (Figure 1b-2). If the missing fact(s) are available, the rule together with its supplemented relevant facts are then sent to Derivation Module to re-attempt derivation. If not, the branch is marked as dead end.

Verification Module. After each successful derivation, this module evaluates the derived fact d_i against the hypothesis $\mathcal{H}$ to determine if the proof is complete. If the derived fact is equivalent to or directly contradicts the hypothesis, the proof is concluded; otherwise, proof exploration continues.

3.3 LLM-free Premise Prioritization

We introduce two heuristics leveraging `spaCy` for premise (fact and rule) prioritization, which provide low-computation and interpretable strategies to improve proof-finding efficiency.

Fact (root) ranking for global ordering of tree search. In LogicTree framework, each given fact f_i serves as the root of a tree, and trees are explored sequentially until the proof is found. As shown in Figure 1a, we first apply a semantic alignment step to prioritize facts that have the same *subject* with the hypothesis $\mathcal{H}$ as tree roots. To further rank facts, we define **cumulative connectivity** between a fact f_i and the rule set $\mathcal{R}$, which is the sum of **semantic overlap** between f_i and each rule $r_i \in \mathcal{R}$. It approximates how many reasoning branches the root fact can initiate through its relevant rules. Facts with zero connectivity are discarded, as they cannot contribute to any derivation. Facts with higher connectivity are prioritized for opening more reasoning paths and higher likelihood of proof discovery in earlier-explored trees. We conduct subject align-

ment and compute semantic overlap using `spaCy`'s efficient dependency parsing.

Rule ranking at local level for early stopping. After each Forward Selection, the selected rules are ranked based on each rule's **semantic overlap** with hypothesis $\mathcal{H}$. This prioritization directs Derivation Module to first apply the rule r_i whose derivation is most likely to verify hypothesis $\mathcal{H}$, facilitating early stopping. For example, to verify $\mathcal{H}$: "*Kevin is uncomfortable.*", a rule r_i such as: "*If ..., the person is uncomfortable.*" would be prioritized.

The computations for semantic overlap and cumulative connectivity are provided in Appendix A.

3.4 LogicTree Algorithm

We employ iterative depth-first search (DFS) algorithm within LogicTree framework to automate systematic exploration as provided in Algorithm 1.

Initially, the algorithm uses *Verification* to check if hypothesis $\mathcal{H}$ can be directly verified from given facts $\mathcal{F}$. If $\mathcal{H}$ is explicitly confirmed or refuted, the algorithm terminates and returns *True* or *False*, respectively. Otherwise, it proceeds with tree search.

As a preliminary, Fact Repository and Derivation HashMap are initialized, and the given facts are ranked using Fact Ranking heuristic. The algorithm starts with the top-ranked fact, which serves as the root of the first tree. Then, *Forward Selection* is called to select relevant rules, which are subsequently ranked using Rule Ranking heuristic. Along each fact-rule branch, one-step inference is conducted. As shown in Algorithm 2, the inference process encapsulates calls to *Derivation* and, if necessary, *Backward Selection*. *Backward Selection* is triggered when the output of *Derivation* indicates a pseudo dead-end. If *Backward Selection* successfully retrieves the missing fact(s) from Fact Repository, then a secondary query to *Derivation* is performed. Together, this modular process (i) decomposes the search for fact-rule combination, reducing complexity from combinatorial to linear (more analysis in Appendix B and Table 3); (ii) ensures each reasoning step involves exactly one rule and its relevant fact(s), producing at most one derived fact per step; (iii) avoids reasoning stagnation by attempting to resolve pseudo dead-ends.

After each inference, *Verification* evaluates the result to determine whether it concludes the proof, enabling early stopping (Algorithm 3). If the result indicates an underivable or redundant (i.e., already in Fact Repository) outcome, DFS backtracks to explore the next branch. Otherwise, the derived fact d_i is appended to the tree for further expansion.

The next iteration begins from derived fact d_i, with LLM modules reset before the next call. By building each step upon prior derivations, our framework maintains logical coherence. Once a tree is fully explored, the algorithm proceeds to the next tree, using the next ranked fact as the root. To avoid excessively long reasoning, we set an LLM query limit on the reasoning process. If all trees are explored or the query limit is reached (whichever occurs first) without verifying the hypothesis $\mathcal{H}$, the algorithm terminates and returns *Unknown*.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate our framework on five multi-step logical reasoning datasets: **RobustLR** (Sanyal et al., 2022), **PrOntoQA-OOD** (Saparov et al., 2023), **ProofWriter** (Tafjord et al., 2020), **ParaRules** (Clark et al., 2020), **LogicNLI** (Tian et al., 2021). For all examples in our experiments, hypothesis concludes as *True*, *False*, or *Unknown*. More details on datasets are provided in Appendix C.1. (Appendix H shows the extension of our framework to mathematical reasoning dataset.)

Baselines. To compare our framework with existing LLM-based reasoning methods, we select baselines from three categories:

- *Strategic LLM prompting*: **CoT** (Wei et al., 2022) prompts the model to generate intermediate reasoning steps before providing final answers.
- *Modular approaches*: **SI** (Selection-Inference) (Creswell et al., 2023) adpots selection and inference modules for iterative reasoning. **CR** (Cumulative Reasoning) (Zhang et al., 2024) introduces a cumulative process of generating new propositions to reach the answer. **ToT** (Tree-of-Thought) (Yao et al., 2023) leverages tree-search algorithm for deliberate reasoning. **LAMBADA** (Kazemi et al., 2023) develops a backward chaining approach for automated reasoning.
- *RL-trained reasoning models*: **o1-mini** (OpenAI, 2024a) and **o3-mini** (OpenAI, 2025) model.

Models. Our framework places no restrictions on the choice of LLMs. Here, we separately employ GPT-4o-mini, GPT-4o, (Achiam et al., 2023) and Llama-3.3 70B (Dubey et al., 2024) within our framework. We reproduce CoT and other modular approaches using the same models for comparison. Further details on models are in Appendix C.2.

Model	Method	Dataset					Avg.†
		LogicNLI	ParaRules	PrOntoQA-OOD	ProofWriter	RobustLR	
GPT-4o-mini	CoT	38.0	48.3	55.0	51.8	62.1	51.2
	SI	46.0	51.3	72.5	55.3	60.4	55.8
	CR	42.7	54.0	75.0	49.7	70.0	56.1
	LAMBADA	54.7	62.0	75.5	68.0	66.3	65.5
	ToT	51.3	64.3	65.5	70.3	72.9	66.5
	LogicTree	**58.0**	**68.7**	**87.5**	**78.8**	**87.1**	**75.7**
GPT-4o	CoT	51.3	69.0	83.0	73.5	79.6	72.0
	SI	48.0	71.0	91.5	68.0	71.3	70.4
	CR	54.0	75.3	91.5	75.3	76.7	75.5
	LAMBADA	68.0	73.3	93.5	86.7	88.3	81.6
	ToT	69.3	75.0	86.5	91.0	89.2	83.1
	LogicTree	**78.7**	**96.3**	**99.0**	**97.0**	**97.9**	**95.6**
Llama-3.3 70B	CoT	46.7	70.8	88.5	75.5	80.0	73.6
	SI	52.0	74.7	92.5	61.7	76.3	70.6
	CR	53.3	76.7	93.0	73.3	70.8	74.6
	LAMBADA	66.7	78.3	91.0	81.7	87.1	81.1
	ToT	69.0	79.7	90.5	87.7	85.4	83.4
	LogicTree	**74.7**	**92.3**	**97.0**	**95.8**	**97.5**	**93.2**

Table 1: Proof accuracy of different methods across five logical reasoning datasets on GPT-4o-mini, GPT-4o, and Llama-3.3 70B. The highest accuracy in each case is in bold; the second-highest is underlined. Avg.† is calculated as the number of correctly proved examples divided by the total number of examples across all five datasets.

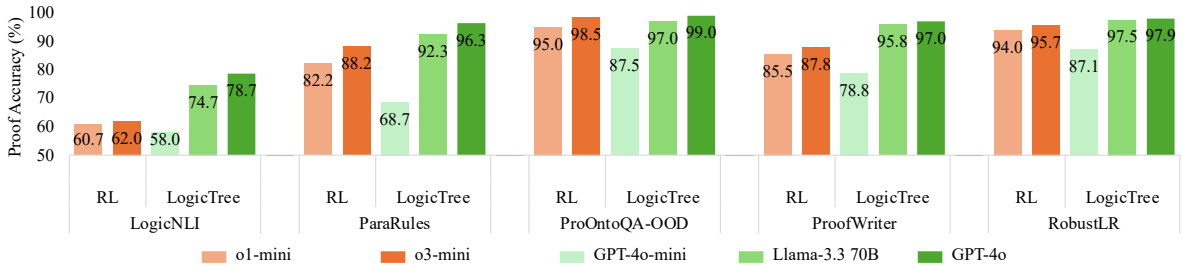


Figure 2: Performance comparison between general LLMs (GPT-4o-mini, GPT-4o, Llama-3.3 70B) applied within LogicTree and RL-trained reasoning models (o1-mini, o3-mini).

Evaluate reasoning accuracy. In logical reasoning, correct label prediction (*True*, *False*, or *Unknown*) does not necessarily indicate correct reasoning, as models may arrive at the correct conclusion through hallucinated premises or spurious correlations (Kazemi et al., 2023; Liu et al., 2023). Similar to Saparov and He (2023), we use **proof accuracy** for rigorous evaluation. We manually verify each example by focusing on the reasoning chain that verifies the hypothesis within the entire reasoning trace. A proof is considered correct if every step in this chain is valid, while the validity of other reasoning paths is disregarded.

4.2 Main Results

As shown in Table 1, our proposed LogicTree consistently outperforms CoT and other modular approaches across all five datasets. Specifically, our method surpasses CoT significantly, with average performance gains of 24.5%, 23.6%, and 19.6% on GPT-4o-mini, GPT-4o, and Llama-3.3 70B, respectively. Compared to ToT, the strongest among other modular methods, our framework achieves average improvements of 9.2%, 12.5%, and 9.8% on the same models. On ParaRules, PrOntoQA-OOD, ProofWriter, and RobustLR datasets, our framework achieves near-perfect proof accuracy with GPT-4o, highlighting its strength in logical reasoning. This strength generalizes across different levels of task difficulty, as shown in Figure 7. Furthermore, when applied within LogicTree, Llama-3.3 70B and GPT-4o outperform RL-trained reasoning models, o1-mini and o3-mini, as shown in Figure 2.

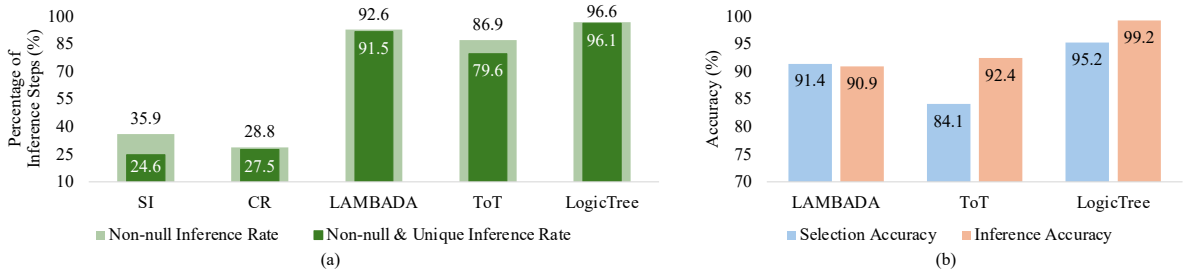


Figure 3: Step-level metrics (§ 5.1) across different methods. (a) Non-null inference rate (the outer bars) and non-null & unique inference rate (the inner bars). (b) Selection accuracy and inference accuracy, evaluated only on tree-based methods. All metrics are manually evaluated on GPT-4o's outputs for 100 examples from ProofWriter.

With Llama-3.3 70B, our method yields 8.7% and 5.2% higher accuracy on average; with GPT-4o, the average gains are 11.1% and 7.6%, respectively.

5 Further Analysis

Figure 6 schematically illustrates how baseline approaches perform logical reasoning with LLMs, which facilitates in-depth performance analysis.

5.1 Factors Impacting Proof Accuracy

To explain the effectiveness of our framework, we define the following step-level metrics:

1. *Non-null Inference Rate*: The percentage of inference steps that result in derived facts.
2. *Non-null & Unique Inference Rate*: The percentage of inference steps that generate new facts (i.e., not previously derived).
3. *Selection Accuracy*: The percentage of selection steps where the selected premises are logically relevant to the parent node during tree expansion.
4. *Inference Accuracy*: The percentage of inference steps that are logically correct given the selected premises.

Logical coherence. Tree-based frameworks (ToT, LAMBADA, LogicTree) exhibit significantly higher performance than SI and CR due to better maintenance of logical coherence. As shown in Figure 6, SI and CR begin each iteration from the updated premise set rather than building directly on prior derivations, disrupting logical coherence. This disruption breaks the continuity of reasoning, resulting in the loss of the logical "pivot" (i.e., prior derivation) needed to guide premise selection. For SI, without this anchor, identifying logically relevant fact-rule combinations becomes difficult, resulting in frequent failed (null) inferences. Additionally, the lack of coherence limits awareness of previous derivations, leading to repeated re-derivation and redundancy. These issues are reflected by SI's low *non-null & unique inference rate* in Figure 3a. CR adopts random combination for premise selection, resulting in an even lower *non-null inference rate* (Figure 3a) due to irrelevant selected premises. Under a fixed iteration budget, failed and redundant steps stall logical progression and ultimately render the proof incomplete.

Premise selection accuracy in tree search. Although ToT builds each reasoning step on prior derivations, it still faces combinatorial search complexity. In conjunctive reasoning scenario (e.g., $f_1 \wedge f_2 \wedge r_1 \Rightarrow d_1$), it requires searching for relevant fact-rule combination ($f_2 \wedge r_1$) for a parent fact node (f_1), making precise selection challenging. Also, to accommodate such search process, ToT does not constrain the number of selected premises per branch (Figure 6d), increasing the risk of selecting irrelevant premises (i.e., distractions). Together, these factors reduce *selection accuracy* and subsequently lead to failed or inaccurate inferences.

Forward vs. Backward tree search strategies. LAMBADA (backward reasoning) starts from the hypothesis and checks each rule to determine its applicability. This method inherently avoids combinatorial search (Table 3), leading to higher *selection accuracy*. However, despite the challenge of combinatorial search, forward reasoning (ToT) achieves higher *inference accuracy* than backward reasoning (Figure 3b). This may be attributed to the prevalence of forward logical flow in pre-training corpus and the autoregressive nature of LLMs, which favors reasoning from premises to conclusions.

Our framework adopts forward reasoning to leverage its aforementioned advantage, while decomposing premise selection to address its search complexity (Table 3), effectively improving both *selection accuracy* and *inference accuracy*, as shown in Figure 3b. Another key reason our framework outperforms ToT is that ToT lacks a mechanism to

leverage derived facts from earlier branches, which may lead to reasoning stagnation (the scenario in Figure 1b-2). Our framework addresses this issue through incorporating Fact Repository (§ 3.2). The impact of this component is evaluated in Table 5.

We further conduct error analysis on CoT, o1-mini, o3-mini, and our framework in Appendix D.

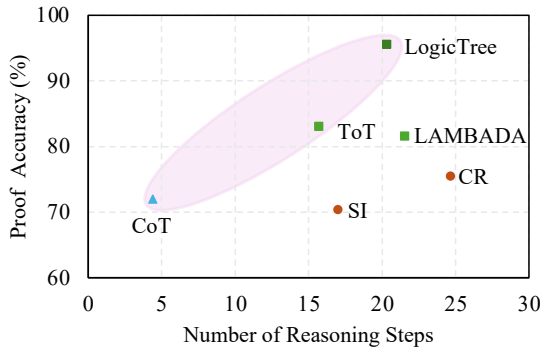


Figure 4: Proof accuracy vs. reasoning steps, averaged across five datasets for GPT-4o. The shaded area illustrates that our framework optimally scales inference-time computation to achieve higher proof accuracy.

5.2 Scaling of Reasoning Length

Proof accuracy vs. reasoning steps. To assess the impact of long reasoning on solving complex logical tasks and compare its effectiveness across different approaches, we measure the average number of reasoning steps for each approach across five datasets (details in Appendix E Table 6). The corresponding proof accuracy and average reasoning steps for each approach are presented in Figure 4. Insufficient reasoning of CoT in complex tasks limits its performance. SI and CR, as analyzed in § 5.1, suffer from high proportion of redundant and failed inferences, which undermine the effectiveness of long reasoning. LAMBADA (backward reasoning) demonstrates more reasoning steps and lower proof accuracy compared to forward reasoning (ToT and LogicTree). Additional analysis comparing forward and backward reasoning is provided in Appendix F Table 4. Compared to ToT, our framework requires more reasoning steps for two main reasons: (1) Our framework decomposes the combinatorial premise search, leading to more steps; (2) In ToT, multiple derivations can occur within a single step (Figure 6d), whereas our framework restricts each step to at most one derivation. The enhanced reasoning granularity ensures strict step-by-step reasoning, optimally increasing reasoning length and leveraging additional inference-time computation to achieve higher proof accuracy, as shown in the shaded areas of Figure 4 and Figure 8.

	ParaRules	ProofWriter	RobustLR
w/o prioritize	17.8 (93.0)	47.0 (91.7)	19.7 (94.6)
LLM-based prioritize	11.6 (95.5)	30.6 (94.8)	17.5 (96.7)
Proposed prioritize	**9.5 (96.3)**	**23.5 (97.0)**	**13.8 (97.9)**

Table 2: Ablation results on GPT-4o: average reasoning steps and proof accuracy (gray, in parentheses).

Premise-prioritization heuristics for efficient scaling. We introduce two premise-prioritization heuristics for strategic proof exploration (§ 3.3). To evaluate their impact on proof search efficiency, we conduct an ablation study across three scenarios: (1) without premise prioritization, where both facts and selected rules are sampled in a random order for exploration; (2) using LLM-based premise prioritization, where two LLM modules are applied: one for fact ranking and one for rule ranking, with details provided in Appendix G; and (3) using our proposed LLM-free heuristics. As shown in Table 2, our proposed heuristics facilitate fewer reasoning steps in proof-finding while attaining higher proof accuracy by avoiding the increased error risk associated with longer reasoning paths.

6 Conclusion

In this work, we propose LogicTree, a novel inference-time modular framework for logical reasoning. Our framework employs algorithm-guided search (DFS) to automate structured exploration while ensuring logical coherence. It incorporates caching mechanism to effectively utilize historical knowledge, preventing reasoning stagnation and minimizing redundancy. Furthermore, we address the combinatorial complexity of premise search and enhance reasoning granularity by restricting inference to at most one derivation per step. This improves stepwise reasoning accuracy and strengthens reasoning rigor. Additionally, we introduce LLM-free heuristics that provide low-computation, explainable strategies to improve proof search efficiency. Experimental results show that LogicTree optimally leverages inference-time scaling to achieve higher proof accuracy, surpassing other modular frameworks and reasoning models, highlighting its strength in logical reasoning.

Limitations

While our framework demonstrates strong performance in logical reasoning tasks, it has some limitations that could open avenues for future work.

First, we evaluate our framework in the domain of logical reasoning, as it represents a distinct type of challenge in reasoning tasks that requires structured and extensive exploration. Our goal is to address this type of reasoning challenge, which often demands more deliberate reasoning. In future work, we plan to extend the framework to more complex domains such as theorem proving.

Second, our framework assumes that all premises (i.e., facts and rules) are explicitly provided. Future work could incorporate premise augmentation with plausible knowledge retrieved from LLM, rather than relying solely on the given premises. Additionally, when facts and rules are not clearly separated, an extra pre-processing step with assistance from LLM may be required (Sun et al., 2024). Also, our premise prioritization strategies rely on simple heuristics. Developing more advanced approaches for proof planning and premise prioritization remains an important direction for future research.

Ethics Statement and Broader Impact

Our work adheres to the Code of Ethics. All utilized methods, models, and datasets are properly cited. The datasets used in our experiments are publicly available, and our research does not involve any private or sensitive information. We confirm that our use of datasets and LLMs aligns with their intended purposes and usage guidelines. A potential risk of our framework lies in the misuse of its outputs in high-stakes domains without sufficient validation or expert review, as LLMs cannot always guarantee fully correct outputs. Nevertheless, when properly applied, our framework contributes to the development of interpretable and automated reasoning systems. Our work has the potential to extend to real-world applications that require rigorous, multi-step decision-making.

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*.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and 1 others. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wangxiang Che. 2025. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*.

Xinyun Chen, Ryan Andrew Chi, Xuezhi Wang, and Denny Zhou. 2024. Premise order matters in reasoning with large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 6596–6620. PMLR.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, and 1 others. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3882–3890. International Joint Conferences on Artificial Intelligence Organization. Main track.

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. *arXiv preprint arXiv:2110.14168*.

Antonia Creswell, Murray Shanahan, and Irina Higgins. 2023. Selection-inference: Exploiting large language models for interpretable logical reasoning. In *The Eleventh International Conference on Learning Representations*.

Bradley Harris Dowden. 2020. *Logical reasoning*. Bradley Dowden.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

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

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Paul Saldyt, and Anil B Murthy. 2024. Position: Llms can't plan, but can help planning in llm-modulo frameworks. In *Forty-first International Conference on Machine Learning*.

Mehran Kazemi, Najoung Kim, Deepti Bhatia, Xin Xu, and Deepak Ramachandran. 2023. LAMBADA: Backward chaining for automated reasoning in natural language. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Toronto, Canada. Association for Computational Linguistics.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. Decomposed prompting: A modular approach for solving complex tasks. In *The Eleventh International Conference on Learning Representations*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, and 1 others. 2024. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. 2023. Transformers learn shortcuts to automata. In *The Eleventh International Conference on Learning Representations*.

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*.

Junjie Liu, Shaotian Yan, Chen Shen, Liang Xie, Wenxiao Wang, and Jieping Ye. 2024. Concise and organized perception facilitates reasoning in large language models. *arXiv preprint arXiv:2310.03309*.

OpenAI. 2024a. Learning to reason with llms.

OpenAI. 2024b. Reasoning best practices.

OpenAI. 2025. Openai o3-mini.

Soumya Sanyal, Zeyi Liao, and Xiang Ren. 2022. RobustLR: A diagnostic benchmark for evaluating logical robustness of deductive reasoners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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*.

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. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

Hongda Sun, Weikai Xu, Wei Liu, Jian Luan, Bin Wang, Shuo Shang, Ji-Rong Wen, and Rui Yan. 2024. DetermLR: Augmenting LLM-based logical reasoning from indeterminacy to determinacy. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Bangkok, Thailand. Association for Computational Linguistics.

Oyvind Tafjord, Bhavana Dalvi Mishra, and Peter Clark. 2020. Proofwriter: Generating implications, proofs, and abductive statements over natural language. *arXiv preprint arXiv:2012.13048*.

Jidong Tian, Yitian Li, Wenqing Chen, Liqiang Xiao, Hao He, and Yaohui Jin. 2021. Diagnosing the first-order logical reasoning ability through LogicNLI. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023a. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *arXiv preprint arXiv:2305.04091*.

Siyuan Wang, Enda Zhao, Zhongyu Wei, and Xiang Ren. 2025. Stepwise informativeness search for improving llm reasoning. *arXiv preprint arXiv:2502.15335*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh 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.

xAI. 2025. Grok 3 beta — the age of reasoning agents.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael Xie. 2023. Self-evaluation guided beam search for reasoning. *Advances in Neural Information Processing Systems*, 36:41618–41650.

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*.

Jundong Xu, Hao Fei, Meng Luo, Qian Liu, Liangming Pan, William Yang Wang, Preslav Nakov, Mong-Li Lee, and Wynne Hsu. 2024a. Aristotle: Mastering logical reasoning with a logic-complete decompose-search-resolve framework. *arXiv preprint arXiv:2412.16953*.

Jundong Xu, Hao Fei, Liangming Pan, Qian Liu, Mong-Li Lee, and Wynne Hsu. 2024b. Faithful logical reasoning via symbolic chain-of-thought. *arXiv preprint arXiv:2405.18357*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. 2025. Demystifying long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2502.03373*.

Yifan Zhang, Jingqin Yang, Yang Yuan, and Andrew C Yao. 2024. Cumulative reasoning with large language models.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

A Computation for Semantic Overlap and Cumulative Connectivity

For each fact f, rule r, as well as the hypothesis $\mathcal{H}$, we extract a structured triple:

$$(\mathcal{S}^f_{\text{Subj}}, \mathcal{S}^f_{\text{Pred}}, \mathcal{S}^f_{\text{SP}}) \quad \text{from a fact,}$$
$$(\mathcal{S}^r_{\text{Subj}}, \mathcal{S}^r_{\text{Pred}}, \mathcal{S}^r_{\text{SP}}) \quad \text{from a rule,}$$
$$(\mathcal{S}^{\mathcal{H}}_{\text{Subj}}, \mathcal{S}^{\mathcal{H}}_{\text{Pred}}, \mathcal{S}^{\mathcal{H}}_{\text{SP}}) \quad \text{from the hypothesis.}$$

In each triple:

- $\mathcal{S}_{\text{Subj}}$ denotes the *Set of Subjects*,
- $\mathcal{S}_{\text{Pred}}$ denotes the *Set of Predicates*,
- $\mathcal{S}_{\text{SP}}$ denotes the *Set of Subject-Predicate pairs* identified via parent-child relations in `spaCy`.

These sets are extracted using `spaCy`'s dependency parser. We represent them as sets to account for the possibility of multiple subjects and predicates within a single fact, rule, or hypothesis.

The **semantic overlap** $Sem(f, r)$ between a fact f and a rule r is defined as:

$$\begin{aligned} Sem(f,r) = {} & 0.25 \cdot \mathbb{I}(\mathcal{S}^f_{\text{Subj}} \cap \mathcal{S}^r_{\text{Subj}} \neq \emptyset) \\ & + \ 0.25 \cdot \mathbb{I}(\mathcal{S}^f_{\text{Pred}} \cap \mathcal{S}^r_{\text{Pred}} \neq \emptyset) \\ & + \ 0.5 \cdot \mathbb{I}(\mathcal{S}^f_{\text{SP}} \cap \mathcal{S}^r_{\text{SP}}) \neq \emptyset), \end{aligned} \quad (1)$$

and similarly, the **semantic overlap** $Sem(r, \mathcal{H})$ between a rule r and hypothesis $\mathcal{H}$ is defined as:

$$\begin{aligned} Sem(r,\mathcal{H}) = {} & 0.25 \cdot \mathbb{I}(\mathcal{S}^r_{\text{Subj}} \cap \mathcal{S}^{\mathcal{H}}_{\text{Subj}} \neq \emptyset) \\ & + \ 0.25 \cdot \mathbb{I}(\mathcal{S}^r_{\text{Pred}} \cap \mathcal{S}^{\mathcal{H}}_{\text{Pred}} \neq \emptyset) \\ & + \ 0.5 \cdot \mathbb{I}(\mathcal{S}^r_{\text{SP}} \cap \mathcal{S}^{\mathcal{H}}_{\text{SP}} \neq \emptyset), \end{aligned} \quad (2)$$

where $\mathbb{I}(\cdot)$ is an indicator function:

$$\mathbb{I}(\text{condition}) = \begin{cases} 1, & \text{if condition is true} \\ 0, & \text{otherwise} \end{cases}$$

We set the coefficients as 0.25, 0.25, and 0.5 respectively, such that the semantic overlap is upper-bounded by 1, achieved when all three conditions are satisfied. Partial (i.e, subject or predicate) overlaps are assigned with non-zero coefficient (0.25) because they may still indicate logical relevance. For example, in the case where the fact is *"Dave is hungry."* and the rule is *"If someone is hungry, they are uncomfortable."*, only the predicate overlaps, but the fact is logically connected to the rule.

The **cumulative connectivity** $\mathcal{C}(f, \mathcal{R})$ between a fact f and the entire rule set $\mathcal{R}$ is defined as the sum of its semantic overlap with each rule in $\mathcal{R}$, i.e.,

$$\mathcal{C}(f, \mathcal{R}) = \sum_{r \in \mathcal{R}} Sem(f, r). \quad (3)$$

A higher cumulative connectivity value indicates that the fact f is likely to initiate more reasoning branches through its relevant rules.

B Linear Premise Search in LogicTree

In our framework, premise search is simplified by decomposing it into forward (rule) selection and backward (fact) selection (§ 3.2), resulting in a linear rather than combinatorial search process.

During forward selection, the framework takes a fact as an anchor and identifies all relevant rules. Although multiple rules may be retrieved in a single LLM query, LLM can perform a process analogous to a linear iteration over the rule set, evaluating each rule independently for relevance without requiring joint combinations.

Similarly, in backward selection, we consider a general conjunctive reasoning case (e.g., $f_1 \wedge f_2 \wedge ... \wedge f_n \wedge r_1 \Rightarrow d_1$), where an anchor fact f_1 partially satisfies a rule r_1. Once this rule r_1 is identified, the remaining required facts (f_2 through f_n) are identified from the rule's conditions and subsequently checked for existence in Fact Repository by Backward Selection Module. LLM can implement this step in a way that resembles a linear scan by verifying the existence of each required fact individually.

In contrast, CoT and ToT require combinatorial search for fact–rule combinations, where the facts and rules must be jointly selected and logically relevant to each other as shown in Table 3, thereby increasing the complexity of LLM's search process.

C Experimental Details

C.1 Dataset

We evaluate on five English-language logical reasoning datasets, as detailed below:

RobustLR (Sanyal et al., 2022) includes Logical Contrast and Logical Equivalence sets for testing the logical robustness on conjunctive, disjunctive, and contrapositive reasoning. We randomly sample 240 examples from the test set.

PrOntoQA-OOD (Saparov et al., 2023) is a synthetic question-answering dataset using fictional names. For evaluation, we use the most challenging 4-hop subset. We randomly sample 200 examples from the test set.

ProofWriter (Tafjord et al., 2020) is a commonly used benchmark for deductive logical reasoning. We evaluate the open-world assumption (OWA) subset, focusing on the hardest depth-5 subset. We randomly sample 600 examples from the test set.

ParaRules (Clark et al., 2020) paraphrases data from ProofWriter into more natural language using crowdsourcing, enhancing text diversity and naturalness. We randomly sample 600 examples from the test set.

LogicNLI (Tian et al., 2021) is the most challenging dataset, featuring a large premise space and numerous reasoning paths, only one of which leads to the proof. We randomly sample 150 examples from the test set.

Few-shot demonstrations for each LLM module are sampled from the training set of each dataset. An example of each dataset is shown in Appendix I.

C.2 Models

Here are the versions of OpenAI's models:
GPT-4o-mini: `gpt-4o-mini-2024-07-18`
GPT-4o: `gpt-4o-2024-08-06`
o1-mini: `o1-mini-2024-09-12`
o3-mini (medium): `o3-mini-2025-01-31`

All OpenAI models are accessed through OpenAI API[2]. Llama-3.3 70B is accessed through Together AI API[3]. We set the temperature to 0.1 for all experiments to encourage more deterministic generation. All results are obtained from a single run. We utilize the Microsoft Guidance library in our implementation[4].

The version of `spaCy` model used in our framework is `en_core_web_lg-3.8.0` (382 MB).

D Error Analysis on CoT, o1-mini, o3-mini, LogicTree

We manually conduct error analysis on CoT, o1-mini, o3-mini, and our framework using ProofWriter dataset. For CoT, we randomly sample 100 failed proofs, while for o1-mini, o3-mini, and our framework, we analyze all failed cases. We categorize the errors into three types: (1) insufficient exploration, (2) wrong derivation, and (3) hallucinated premise. The proportion of these error types for each method, along with illustrative examples, is shown in Figure 9. Our framework exhibits significantly fewer errors caused by insufficient exploration. In addition, our framework does not suffer from hallucinated premises, as access to the hypothesis is restricted to Verification Module only. This prevents the generation of unsupported premises that favor verifying the hypothesis during premise selection and inference.

Our findings on why CoT struggles with complex logical reasoning align with prior research: (1) it faces difficulty when premises are unordered and contain distractions (Chen et al., 2024), and (2) it lacks systematic exploration when reasoning requires navigating extensive branching (Saparov and He, 2023). The high branching factor that complicates exploration, along with sensitivity to distractions, also limits the performance of o1-mini and o3-mini in complex logical reasoning compared to their effectiveness in coding and math. To address this, a modular method for precise premise selection which strengthens robustness to distractions, combined with an algorithm-guided approach for systematic proof searching, provides a promising foundation. Our framework builds upon and extends these components, addressing their limitations to develop a logically complete algorithm that enables rigorous and coherent reasoning, ultimately achieving superior proof accuracy.

E Number of Reasoning Steps, Generated Tokens, and Inference Time

The following elaborates on how we measure the number of reasoning steps for each approach.

(1) For CoT, we define one reasoning step as a combination of premise selection and an inference based on the selected premises. To make step counting explicit in LLM's output, we number each reasoning step in few-shot demonstrations (Figure 17).

(2) For SI, we set the maximum number of iterations to 10, as we find the framework typically fails to generate new derivations beyond this point. Each iteration consists of one query to LLM selection module and one query to LLM inference module. The process terminates early if the hypothesis is successfully verified. We define the total number of reasoning steps in SI as the total number of LLM module queries made across the iterations.

(3) For CR, we use the framework's default hyperparameters for reasoning. The total number of reasoning steps in CR is calculated as the total number of LLM module queries made during the iterations, plus the number of steps in the final CoT reasoning process.

(4) For ToT, LAMBADA, and LogicTree, each

[2] https://platform.openai.com/docs/overview
[3] https://www.together.ai/models/llama-3-3-70b
[4] https://github.com/guidance-ai/guidance

query to an LLM module is counted as one reasoning step. We set the step limit L (in Algorithm 1) to 80 for all methods. Tree search terminates early if the hypothesis is successfully verified.
(5) Since reasoning with o1-mini and o3-mini includes unobservable intermediate steps, we exclude them from the analysis of reasoning steps.

Table 6 shows the average number of reasoning steps across five logical reasoning datasets for different methods. Table 7 presents the average number of generated tokens and inference time for those different methods. The number of generated tokens is obtained using `completion_tokens` from the completion response.

F Performance Analysis: Forward vs. Backward Reasoning

In Table 4, we present in-depth analysis to explain why backward reasoning requires more reasoning steps than forward reasoning in iterative tree search, based on three key factors: (1) utilization of historical knowledge; (2) evaluation of hypothesis and its negation; and (3) branching complexity in reasoning paths.

The lower proof accuracy of LAMBADA (backward reasoning) compared to forward reasoning methods (ToT and LogicTree) can be explained by two main reasons: (1) LLMs demonstrate higher stepwise inference accuracy in forward reasoning as shown in Figure 3b (§ 5.1); (2) the larger number of reasoning steps in backward reasoning increases the likelihood of making errors.

G Using LLM Modules for Premise Prioritization

The prompts used for the LLM modules in fact and rule ranking are provided in Appendix I. We do not apply predefined criteria in these prompts, allowing us to assess LLM's inherent ability on premise prioritization. For the average reasoning steps (*LLM-based prioritize*) reported in Table 2, we do not include LLM queries related to fact and rule ranking. Even without this overhead, LLM-based premise prioritization results in more reasoning steps than our LLM-free heuristics. This reflects the limitation in LLM-based proof planning. Successfully guiding the reasoning process requires the model to already have an accurate understanding of how to reach the proof in advance, yet this ability is not evidenced by the limited performance of CoT (Table 1). Based on the results in Table 2, our simple LLM-free heuristics prove effective for strategic proof exploration.

H Extension to Mathematical Reasoning

LogicTree is primarily designed to strengthen the logical reasoning capabilities of LLMs. Beyond its original focus, it can be readily adapted to other types of reasoning tasks that start from a given set of information. Our framework systematically combines relevant information for derivation, leverages derived information, and facilitates structured problem solving. In Figure 10, we illustrate how LogicTree performs mathematical reasoning on an example from GSM8K (Cobbe et al., 2021). We further compare the accuracy of CoT, ToT, and LogicTree on a subset of 300 randomly selected examples from GSM8K test set. All methods are evaluated on Qwen2.5-7B (Yang et al., 2024), with LogicTree demonstrating superior performance.

I Dataset Example and Prompt for LLM Modules

Figure 11, Figure 12, Figure 13, Figure 14, Figure 15 show an example of LogicNLI, PrOntoQA-OOD, ProofWriter, RobustLR, ParaRules, respectively.

Figure 16 shows the prompts for reasoning with OpenAI's o1-mini and o3-mini model. We use the same prompts for all the five datasets.

Figure 17 shows the prompts for chain-of-thought. The instructions in the prompt are identical across all five datasets, while the demonstrations are sampled from the training set of each respective dataset. We use examples from ProofWriter as illustrations.

Figure 18, Figure 19, Figure 20, Figure 21 separately show the prompts for Forward Selection Module, Backward Selection Module, Derivation Module, Verification Module in our framework. The instructions in the prompt are consistent across all five datasets, while the demonstrations are sampled from the training set of each respective dataset. We use examples from ProofWriter as illustrations.

Figure 22 and Figure 23 respectively show the prompts for Fact Ranking Module and Rule Ranking Module, which are used in the ablation studies with results presented in Table 2.

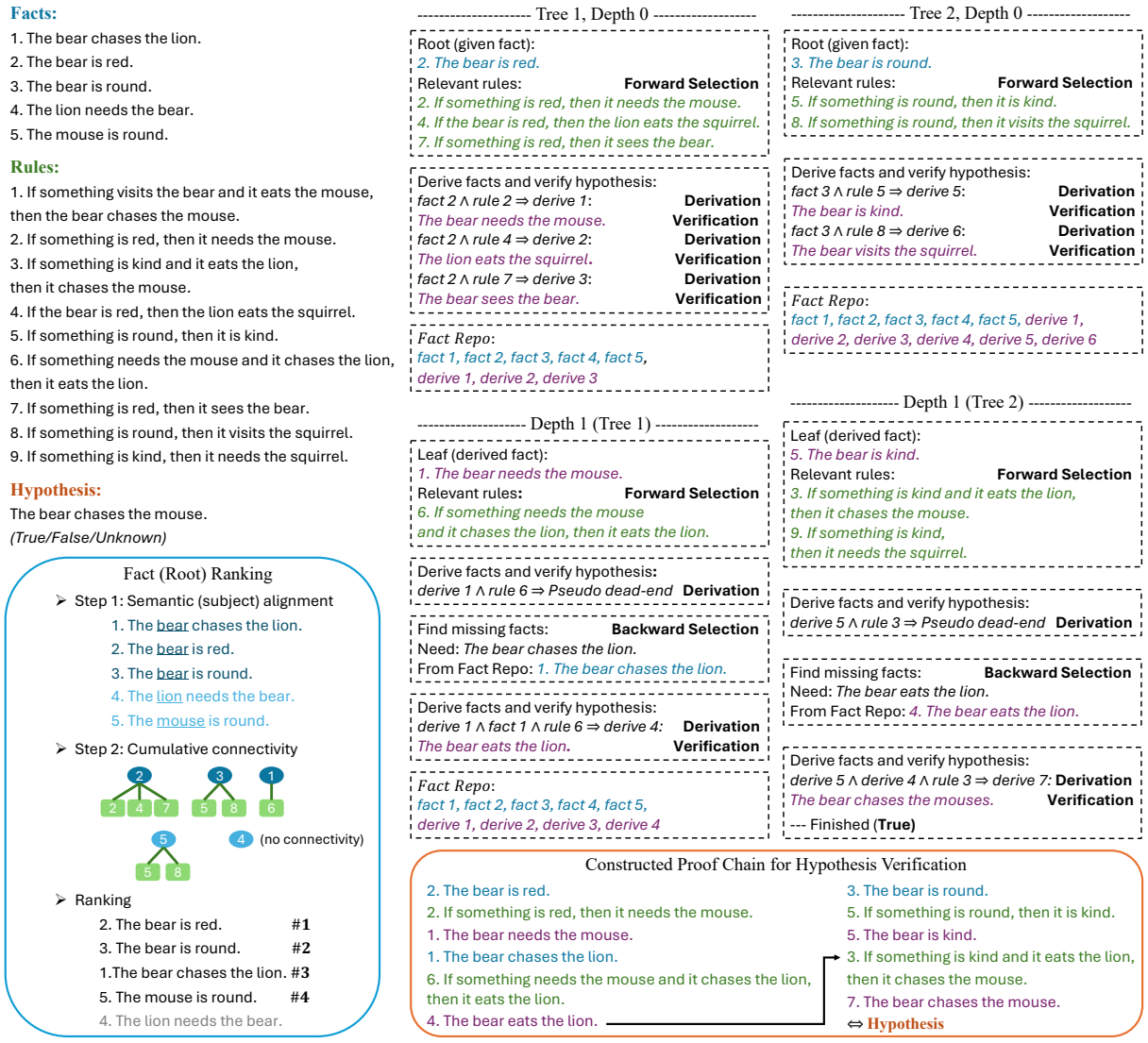
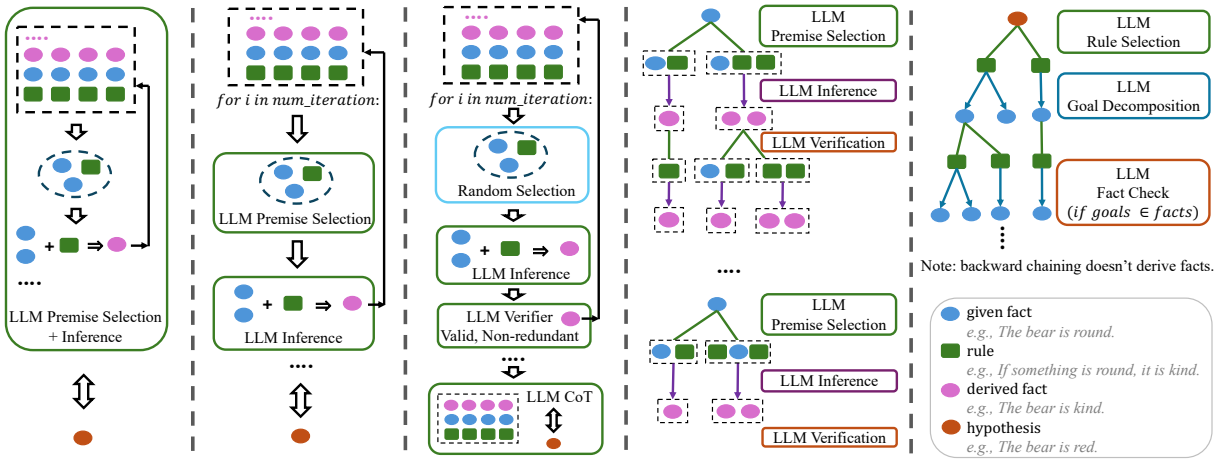


Figure 5: The proof exploration trace of LogicTree on an example from ProofWriter (Tafjord et al., 2020). Rule Ranking, Derivation HashMap and some leaves (*derive2, derive3, derive6*) are omitted for brevity.



(a) Chain-of-Thought (b) Selection-Inference (c) Cumulative Reasoning (d) Tree-of-Thought (e) LAMBADA (backward chaining)

Figure 6: Schematic illustrations of baseline approaches for logical reasoning with LLMs: (a) Chain-of-Thought; (b) Selection-Inference; (c) Cumulative Reasoning; (d) Tree-of-Thought; (e) LAMBADA.

Algorithm 1 LogicTree DFS Algorithm

Require:
Premises $\mathcal{F} \cup \mathcal{R}$, Hypothesis $\mathcal{H}$
Large language model LLM, NLP library `spaCy`

$s \leftarrow 0$ ▷ Number of reasoning steps
$step\_limit \leftarrow L$
for each $f_i \in \mathcal{F}$ **do**
  $result \leftarrow \texttt{Early\_Stop}(LLM, f_i, \mathcal{H}, s)$
  if $result \neq$ Unknown **then**
    return $result$ ▷ Verified
  end if
end for
$Fact\_Repo \leftarrow \mathcal{F}$
$Derivation\_HashMap \leftarrow \{\}$
$\mathcal{F}_{rank} \leftarrow \texttt{Fact\_Ranking}(\texttt{spaCy}, \mathcal{F}, \mathcal{R})$
for each $f_i \in \mathcal{F}_{rank}$ **do**
  if $s > step\_limit$ **then**
    break
  end if
  $Tree \leftarrow [f_i]$ ▷ Initiate the root of a tree
  while $Tree \neq \emptyset$ **do**
    $f_l \leftarrow Tree.\texttt{pop()}$ ▷ LIFO, get the leaf fact
    $\mathcal{R}^{relv} \leftarrow \texttt{Forward\_Selection}(LLM, f_l, \mathcal{R}, s+1)$
    $\mathcal{R}^{relv}_{rank} \leftarrow \texttt{Rule\_Ranking}(\texttt{spaCy}, \mathcal{R}^{relv}, \mathcal{H})$
    for each $r_i \in \mathcal{R}^{relv}_{rank}$ **do**
      $d_i \leftarrow \texttt{Inference}(LLM, f_l, r_i, Fact\_Repo, s)$
      $result \leftarrow \texttt{Early\_Stop}(LLM, d_i, \mathcal{H}, s)$
      if $result \neq$ Unknown **then**
        return $result$ ▷ Verified
      end if
      if $(d_i \neq \text{null}) \wedge (d_i \notin Fact\_Repo)$ **then**
        $Tree.\texttt{append}(d_i)$ ▷ Extend branches
        $Fact\_Repo \leftarrow Fact\_Repo \cup \{d_i\}$
        $Derivation\_HashMap[d_i] \leftarrow path_{d_i}$
      end if
    end for
  end while
end for
return Unknown

Algorithm 2 Inference Function

Require:
$f_l, r_i, Fact\_Repo, s$
Large language model LLM

$d_i \leftarrow \texttt{Derivation}(LLM, f_l, r_i, s+1)$
if $d_i \neq$ null **then**
  return d_i
end if
▷ Use textual pattern matching
if $\texttt{PseudoDeadEnd}(d_i)$ **then**
  ▷ Search missing fact
  $f_s \leftarrow \texttt{Backward\_Selection}(LLM, f_l, r_i, Fact\_Repo, s+1)$
  if $f_s \neq$ null **then**
    $d'_i \leftarrow \texttt{Derivation}(LLM, f_l \wedge f_s, r_i, s+1)$
    return d'_i
  end if
end if
return null

Algorithm 3 Early_Stop Function

Require:
f_i or d_i, $\mathcal{H}$, s
Large language model LLM

$result \leftarrow \texttt{Verification}(LLM, f_i \text{ or } d_i, \mathcal{H}, s+1)$
if $result \neq$ Unknown **then**
  ▷ $\mathcal{H}$ is verified
  ▷ $result$ is either *True* or *False*
  return $result$
end if
▷ $\mathcal{H}$ is not verified
return Unknown

Facts $\mathcal{F}$ + Rules $\mathcal{R}$:

[...] The bald eagle likes the dog. The bald eagle needs the tiger. The bald eagle sees the tiger. The bald eagle needs the dog. The dog is blue. The dog sees the tiger. The rabbit is green. The tiger needs the bald eagle. [...]
[...]
If someone needs the bald eagle and the bald eagle sees the tiger then they are rough.
If someone needs the dog and they like the dog then they like the tiger.
If someone likes the bald eagle then the bald eagle needs the dog.
If someone is rough and they like the dog then the dog needs the tiger.
[...]

Hypothesis:

The bald eagle likes the tiger.

► CoT

Premise Search

The bald eagle likes the dog.

The bald eagle needs the dog.

If someone needs the dog and they like the dog then they like the tiger.

One-step search

Combinatorial complexity

► ToT

Parent node (anchor)

📍 The bald eagle likes the dog.

Premise Search

The bald eagle needs the dog.

If someone needs the dog and they like the dog then they like the tiger.

One-step search

Combinatorial complexity

► LAMBADA

Parent node (anchor)

📍 The bald eagle likes the tiger.

Premise (rule) Search

If someone needs the dog and they like the dog then they like the tiger.

Fact Check (if $f \in \mathcal{F}$)

The bald eagle needs the dog.

The bald eagle likes the dog.

One-step search + one-step check

Linear complexity

► LogicTree

Parent node (anchor)

📍 The bald eagle likes the dog.

Forward (rule) Selection

If someone needs the dog and they like the dog then they like the tiger.

fact-rule pair (anchor)

📍 The bald eagle likes the dog. + If someone needs the dog and they like the dog then they like the tiger.

Backward (fact) Selection

The bald eagle needs the dog.

Two-step search

Linear complexity

Table 3: Complexity of premise search in CoT, ToT, LAMBADA, and our framework. An example involving *conjunctive reasoning* from ProofWriter is used for analysis. Note that although ToT's complexity becomes linear in non-conjunctive cases, the reasoning type is not known beforehand. Therefore, ToT must retain its higher search complexity in the general case to accommodate complex scenarios.

Factor	Forward Reasoning	Backward Reasoning
Utilization of historical knowledge (intermediate derivations)	(i) *Immediate utilization:* Derived facts are immediately valid and can be utilized across reasoning chains through caching, enabling efficient information sharing and minimizing redundant computation. (ii) *Dead ends still contribute:* Even if a reasoning chain fails to reach the final hypothesis, its intermediate derived facts are valid and can be utilized to support other inference paths.	(i) *Delayed utilization:* Intermediate facts are only validated upon completing a successful reasoning chain and cannot be utilized across other chains while the current chain is still in progress. (ii) *Dead ends yield nothing:* If a reasoning chain fails, none of its intermediate facts are proven valid and therefore cannot be utilized in other inference paths.
Evaluation of hypothesis and its negation	Forward reasoning starts from the given premises and derives logically valid conclusions, using those results to determine the truth value of the hypothesis. The derivation process does not explicitly attempt to prove or disprove the hypothesis.	Backward reasoning must start from and evaluate both the hypothesis and its negation to determine the truth value, as failure to prove one does not imply the truth of the other—due to the possibility of an *unknown* outcome. This requirement increases the overall reasoning steps. E.g., to evaluate the hypothesis Dave is blue, backward reasoning explores both the supporting path starting with If ..., then Dave is blue, and the opposing path starting with If ..., then Dave is not blue.
Branching complexity in reasoning paths	Forward reasoning applies rules to known facts independently. Path count grows **additively**. E.g., given the facts: Dave is blue and Dave is cold, suppose there are: m rules of the form: If Dave is blue, then ... n rules of the form: If Dave is cold, then ... The total number of paths to be explored is $m + n$.	Backward reasoning explores all combinations of rules whose conclusions match intermediate goals. Path count grows **multiplicatively**. E.g., to satisfy the conjunction of goals: Dave is blue and Dave is cold, suppose there are: m rules of the form: If ..., then Dave is blue. n rules of the form: If ..., then Dave is cold. The total number of paths to be explored is $m * n$.

Table 4: Analysis of three key factors that lead to more **reasoning steps** in backward reasoning compared to forward reasoning in *iterative tree search* for logical reasoning tasks: (1) Utilization of historical knowledge, (2) Evaluation of hypothesis and its negation, and (3) Branching complexity in reasoning paths. Together, these factors result in a higher average number of reasoning steps in backward reasoning, as illustrated in Figure 4 and Figure 8.

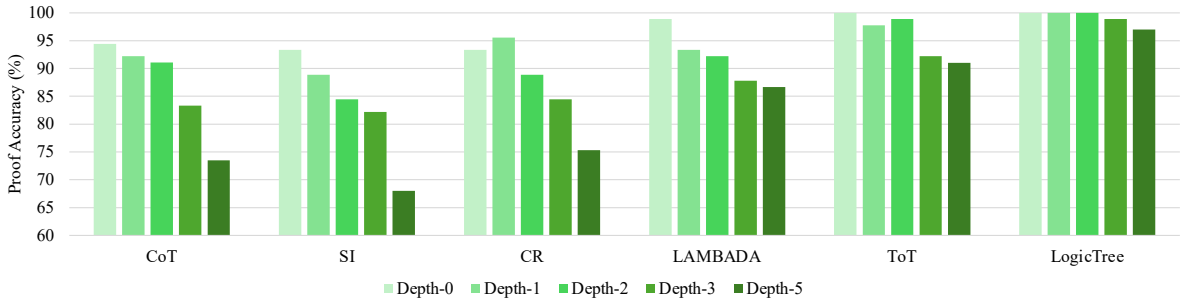
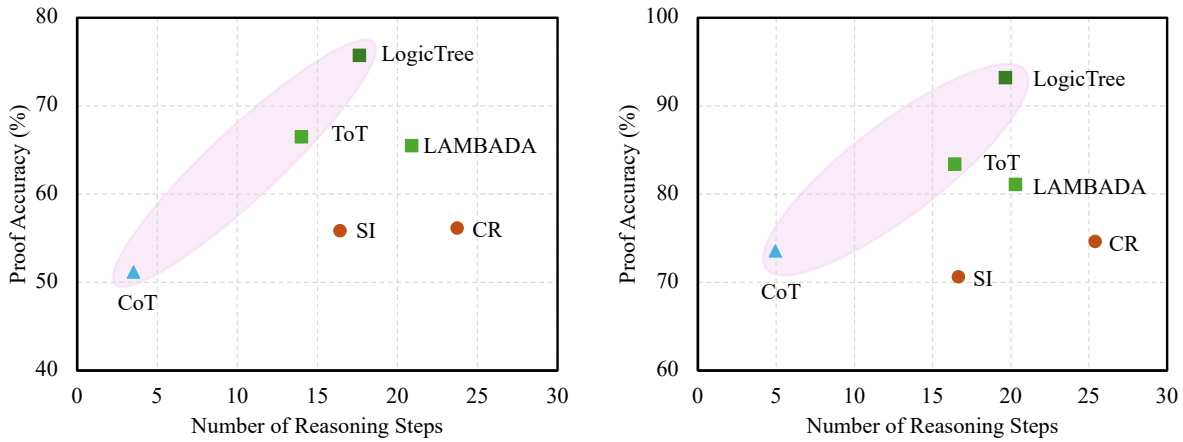


Figure 7: Proof accuracy of different methods across different levels of task difficulty (measured by depth) in ProofWriter on GPT-4o. LogicTree consistently outperforms other methods at all depths, demonstrating superior robustness to problem difficulty.

	LogicNLI	ParaRules	PrOntoQA-OOD	ProofWriter	RobustLR
• ***Llama-3.3 70B***					
LogicTree w/o fact repo	68.7	81.7	90.0	88.7	92.1
LogicTree	**74.7**	**92.3**	**97.0**	**95.8**	**97.5**
• ***GPT-4o***					
LogicTree w/o fact repo	72.0	87.5	88.0	90.0	93.8
LogicTree	**78.7**	**96.3**	**99.0**	**97.0**	**97.9**

Table 5: Impact of the fact repository (fact repo) on proof accuracy across five datasets within our framework.



(a) GPT-4o-mini (b) Llama-3.3 70B

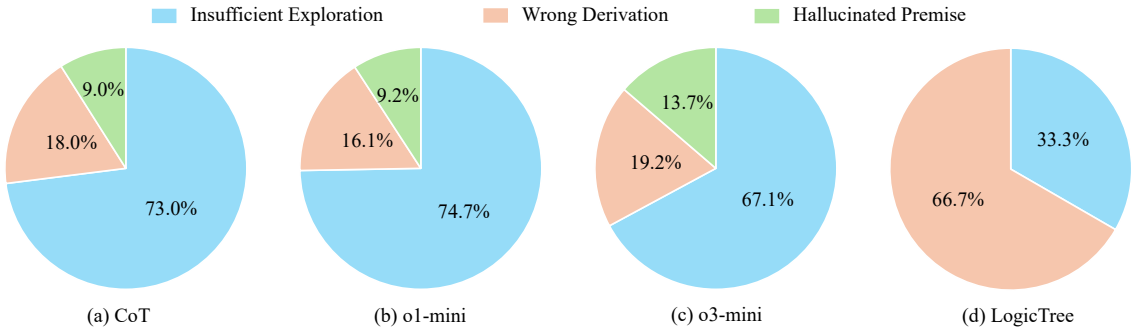
Figure 8: Proof accuracy vs. reasoning steps, averaged across five datasets for (a) GPT-4o-mini and (b) Llama-3.3 70B. The shaded area illustrates that our framework optimally scales inference-time computation to achieve higher proof accuracy.

Model	Method	Dataset					Avg.†
		LogicNLI	ParaRules	PrOntoQA-OOD	ProofWriter	RobustLR	
GPT-4o-mini	CoT	3.4	3.0	3.7	3.5	4.8	3.5
	SI	16.6	18.9	8.6	18.5	11.7	16.4
	CR	25.7	23.1	22.1	23.3	26.7	23.8
	LAMBADA	14.0	23.7	22.6	18.0	24.0	20.9
	ToT	36.2	5.1	21.7	17.0	8.5	14.0
	LogicTree	43.2	6.8	28.2	21.2	11.1	17.6
GPT-4o	CoT	5.2	3.5	3.8	4.3	6.6	4.4
	SI	19.6	17.0	10.7	19.4	14.6	17.0
	CR	24.1	25.1	21.8	25.7	23.4	24.6
	LAMBADA	22.9	21.2	22.5	19.6	25.6	21.5
	ToT	32.9	6.4	23.1	20.2	10.3	15.7
	LogicTree	45.6	9.5	32.2	23.5	13.8	20.3
Llama-3.3 70B	CoT	6.3	3.5	4.0	5.6	6.8	4.9
	SI	11.7	21.5	3.8	21.6	6.2	16.7
	CR	27.1	24.6	22.2	27.3	24.2	25.4
	LAMBADA	19.6	25.2	17.9	14.0	26.4	20.3
	ToT	31.3	7.4	23.1	21.2	12.3	16.4
	LogicTree	39.1	7.3	30.6	25.6	14.4	19.7

Table 6: Average number of reasoning steps for different methods across five logical reasoning datasets on GPT-4o-mini, GPT-4o, and Llama-3.3 70B. Avg.† is calculated as the total number of reasoning steps divided by the total number of examples across all five datasets.

Method	Dataset									
	LogicNLI		ParaRules		PrOntoQA-OOD		ProofWriter		RobustLR	
	Token #	Time (min)	Token #	Time (min)	Token #	Time (min)	Token #	Time (min)	Token #	Time (min)
CoT	145.4	0.11	105.5	0.08	125.9	0.09	103.2	0.08	171.9	0.12
SI	681.7	0.80	688.3	0.79	233.2	0.28	490.5	0.68	502.1	0.60
CR	1231.6	1.43	871.6	1.04	490.4	0.62	816.7	0.94	800.7	0.97
LAMBADA	1181.0	1.36	758.6	0.97	566.3	0.67	550.5	0.74	921.4	1.07
ToT	1896.7	2.13	258.4	0.30	568.2	0.70	709.3	0.86	355.6	0.42
o1-mini	3253.3	1.11	464.5	0.19	943.0	0.39	953.1	0.42	534.8	0.24
o3-mini	2412.3	1.02	589.2	0.27	683.0	0.34	849.1	0.36	481.9	0.21
LogicTree	3627.0	4.07	440.6	0.54	1082.0	1.47	738.4	0.89	493.9	0.61

Table 7: Average number of generated tokens and inference time (in minutes) for different methods across five logical reasoning datasets. CoT, SI, CR, LAMBADA, ToT, and LogicTree results are based on GPT-4o.



(a) CoT (b) o1-mini (c) o3-mini (d) LogicTree

An example of Insufficient Exploration

Facts: The bald eagle chases the cat. The bald eagle is kind. The bald eagle sees the dog. The cat is rough. The dog is blue. The dog is red. The dog is rough. The dog sees the cat. The dog does not see the lion. The lion sees the bald eagle.

Rules: If something chases the cat then the cat chases the dog. If something chases the dog then it eats the bald eagle. If something is kind and it sees the lion then it is not red. If something chases the dog then the dog chases the bald eagle. If the dog eats the bald eagle then the dog sees the bald eagle. If something is cold then it sees the lion. If something sees the cat and it is blue then the cat is kind. If something is blue and it chases the bald eagle then the bald eagle is cold.

Hypothesis: The cat eats the bald eagle. (True)

Inference: 1. The bald eagle chases the cat. According to the rule, if something chases the cat, then the cat chases the dog. So, the cat chases the dog.
2. There is no rule that directly or indirectly states that if the cat chases the dog, then the cat eats the bald eagle.
3. None of the given rules or statements indicate that the cat eats the bald eagle.
4. Therefore, based on the provided information, it is unknown whether the cat eats the bald eagle.

An example of Wrong Derivation

Facts: Anne is blue. Erin is quiet. Fiona is big. Fiona is furry. Fiona is nice. Harry is furry. Harry is not rough.

Rules: Round people are blue. If something is round and not big then it is not nice. Quiet things are nice. All blue, round people are nice. Nice, rough things are round. If something is furry then it is round. Furry, blue things are quiet. Nice things are furry. If something is round and blue then it is not rough.

Hypothesis: Anne is quiet. (Unknown)

Inference: 1. Anne is blue. 2. Round people are blue, so Anne is round. 3. All blue, round people are nice, so Anne is nice.
4. Nice things are furry, so Anne is furry. 5. Furry, blue things are quiet, so Anne is quiet.
Based on the given information, we can deduce that Anne is quiet.

An example of Hallucinated Premise

Facts: The bald eagle sees the cat. The cat is round. The cat needs the rabbit. The cat sees the lion. The cat visits the lion. The lion is big. The lion is cold. The lion is nice. The lion visits the rabbit. The rabbit is nice. The rabbit is round. The rabbit sees the cat.

Rules: If something is nice then it is big. If something is kind and it sees the rabbit then it needs the lion. If something needs the rabbit then it is nice. All cold, big things are kind. If something sees the bald eagle and the bald eagle is nice then it needs the rabbit. If something sees the lion then it visits the rabbit. If the bald eagle sees the cat then the bald eagle visits the rabbit. If something visits the rabbit then it needs the rabbit. If something is big then it is cold.

Hypothesis: The lion needs the lion. (Unknown)

Inference: 1. The lion is big and cold. According to the rule "All cold, big things are kind," the lion is kind.
2. The lion sees the rabbit. According to the rule "If something is kind and it sees the rabbit then it needs the lion," the lion needs the lion.
Therefore, the statement "The lion needs the lion" is true.

(e) Examples of wrong proof corresponding to the three error types. The erroneous step is highlighted in red.

Figure 9: Error analysis. (a–d) show the proportions of three error types for CoT (on GPT-4o), o1-mini, o3-mini, and LogicTree (on GPT-4o), respectively. (e) Examples of wrong proof corresponding to the three error types.

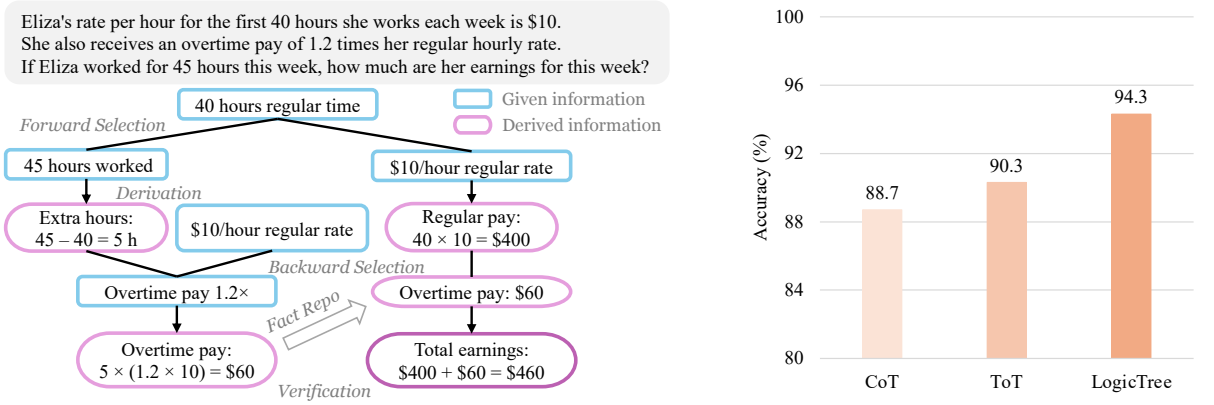


Figure 10: LogicTree on GSM8K. **Left**: Case study illustrating how LogicTree solves a math problem. **Right**: Accuracy comparison among CoT, ToT and LogicTree.

LogicNLI

Premises (Facts + Rules):
Carrick is filthy. Carrick is not financial. Galvin is grieving. Blake is filthy. Oscar is not relieved. Perry is not filthy. Blake is financial. Perry is relieved. Toby is financial. Perry is filthy. Oscar is not filthy. Toby is not filthy.
Someone who is filthy is always unlikely. It can be concluded that Carrick is not unlikely and Galvin is not filthy once knowing that Carrick is relieved and Perry is filthy. If there is at least one people who is both not relieved and filthy, then Blake is grieving. Someone being both filthy and not unlikely is equivalent to being relieved. If Blake is unlikely and Galvin is relieved, then Oscar is filthy. If Perry is relieved, then Carrick is not filthy, and vice versa. Carrick being not grieving or Toby being not filthy implies that Carrick is filthy. If Perry is not filthy or Carrick is not grieving, then Conway is not filthy. If there is at least one people who is not filthy, then Oscar is financial. Someone who is filthy is always both not filthy and not financial. If there is someone who is either not filthy or grieving, then Toby is not filthy. If there is someone who is both not grieving and filthy, then Blake is filthy.

Hypothesis:
Carrick is relieved.

Figure 11: An example of LogicNLI (Tian et al., 2021).

PrOntoQA-OOD

Premises (Facts + Rules):
Rex is a tumpus. Rex is a vumpus. Rex is a lempus. Rex is a lempus. Rex is a wumpus. Rex is a jompus.
Zumpuses are grimpuses. Each dumpus is a gorpus. Everything that is a lempus, a wumpus, and a brimpus is a grimpus, a dumpus, and a zumpus. Each grimpus is an impus. Zumpuses are shumpuses. Grimpuses are gorpuses. Everything that is a lempus and a wumpus and a brimpus is a rompus. Everything that is a tumpus and a lempus and a vumpus is a gorpus. Grimpuses are yumpuses.

Hypothesis:
Rex is an impus.

Figure 12: An example of PrOntoQA-OOD (Saparov et al., 2023).

ProofWriter

Premises (Facts + Rules):
The bald eagle chases the cow. The bald eagle is kind. The bald eagle is rough. The bald eagle needs the rabbit. The cow chases the rabbit. The cow is cold. The cow is green. The cow is red. The rabbit does not chase the bald eagle. The rabbit chases the cow. The rabbit does not eat the bald eagle. The rabbit eats the cow. The rabbit is cold. The rabbit is green. The squirrel eats the cow. The squirrel does not eat the rabbit.
If something needs the bald eagle then the bald eagle chases the rabbit. If the squirrel is rough and the squirrel is not kind then the squirrel is green. If something chases the bald eagle then it needs the squirrel. If something needs the rabbit then it chases the bald eagle. If something chases the cow then the cow eats the bald eagle. If something chases the bald eagle and it does not need the bald eagle then it is red. If something needs the squirrel then the squirrel needs the rabbit.

Hypothesis:
The squirrel needs the rabbit.

Figure 13: An example of ProofWriter (Tafjord et al., 2020).

RobustLR

Premises (Facts + Rules):
Fiona is not Bob's mother. Harry is Charlie's son.
The father of Dave is Bob if Gary is not green. If Fiona is not Bob's mother then Charlie is not Dave's aunt. If Fiona is not Bob's son then Charlie is the aunt of Dave. If Bob is rough then Bob is Dave's daughter. Fiona is not the son of Bob if Bob is rough. Dave is not kind if Fiona is not the son of Bob. If The son of Bob is not Fiona then Harry is not white. If Fiona is Harry's grandfather then Harry is white. If Bob is rough then The grandfather of Harry is not Fiona. Anne is not furry if The aunt of Dave is not Charlie. Bob is not Dave's father if Bob is rough. Gary is green if Bob is rough. The husband of Dave is not Anne if The son of Bob is not Fiona. If Bob is not the daughter of Dave then Gary is not green. Fiona is the grandfather of Harry if The mother of Bob is not Fiona. If Anne is not the husband of Dave then Anne is furry. If Harry is white and The son of Charlie is Harry then The daughter of Dave is Bob.

Hypothesis:
The daughter of Dave is not Bob.

Figure 14: An example of RobustLR (Sanyal et al., 2022).

ParaRules

Premises (Facts + Rules):
Bob is a cold and round man who has red and green skin. Charlie is a kind person and he is also often cold. That guy Eric sure is nice. Harry is a really nice guy with a big round body, usually wearing red.
People who are round and red tend to be rough. If a person acts cold yet nice and green, they will be kind. If you meet someone with rough skin who is cold from being outside, you'll notice they are nice. Every time you meet someone kind and nice, they'll be green, too. Big people with red hair are cold because they cannot find coats that fit. It's a certainty that any green, big and kind individual is going to be nice. A big round young person is often blue.

Hypothesis:
Bob is nice.

Figure 15: An example of ParaRules (Clark et al., 2020).

Prompts for o1-mini & o3-mini

Instructions:
You will do logic reasoning tasks. You will be given a set of premises and a hypothesis. You need to answer if the hypothesis is *True* or *False* or *Unknown* based on the premises.
(The last sentence in the response should be in the format of "Therefore, the hypothesis is True / False / Unknown.")

Query:
Premise:
`query_premise`
Hypothesis:
`query_hypothesis`
Reasoning:
`LLM_output`

Figure 16: The prompts for reasoning with OpenAI's o1-mini and o3-mini model. Following OpenAI's guidance (OpenAI, 2024b), we adopt zero-shot prompting and keep the prompts simple and direct.

Prompts for Chain-of-Thought

Instructions:
Suppose you are one of the greatest AI scientists, logicians. Given some context as premise, the task is to answer if a logical reasoning question (hypothesis) is *True* or *False* or *Unknown*.
Definitions:
- True: The hypothesis can be logically derived from the premises.
- False: Only the negation of the hypothesis can be logically derived from the premises.
- Unknown: Neither the hypothesis nor its negation can be logically derived from the premises.
Let us think step by step.

Demonstrations:
`Example_1`
*# Example's hypothesis is *True**
.....
`Example_2`
*# Example's hypothesis is *False**
Premise:
The lion is cold.
The lion is nice.
The lion likes the squirrel.
The squirrel is round.
The squirrel sees the lion.
The squirrel visits the cow.
The tiger likes the cow.
The tiger likes the squirrel.
If something is cold then it visits the tiger.
If something visits the tiger then it is nice.
If something sees the tiger and it is young then it is blue.
If something is nice then it sees the tiger.
If something likes the squirrel and it likes the cow, then it visits the tiger.
If something is nice and it sees the tiger then it is young.
If the cow is cold and the cow visits the lion then the lion sees the squirrel.
Question:
Based on the above information, is the following statement true, false, or unknown? The tiger is not young.
Reasoning: *# Expected results*
1. The tiger likes the cow. The tiger likes the squirrel. If something likes the squirrel and it likes the cow, then it visits the tiger. So the tiger visits the tiger.
2. If something visits the tiger then it is nice. So the tiger is nice.
3. If something is nice then it sees the tiger. So the tiger sees the tiger.
4. If something is nice and it sees the tiger then it is young. So the tiger is young.
5. It contradicts "The tiger is not young." in the given question. So the answer is False.
`Example_3`
*# Example's hypothesis is *Unknown**
.....
———————

Query:
Premise:
`query_premise`
Question:
`query_hypothesis`
Reasoning:
`LLM_output`

Figure 17: The prompts for chain-of-thought reasoning. We number the reasoning steps in demonstrations to make the step counting explicit in LLM's output. Demonstrations with hypotheses labeled as *True* and *Unknown* are omitted for brevity.

Prompts for Forward Selection Module

Instructions:
Imagine you are one of the greatest AI scientists. You are given **a fact** and **a list of rules** (each rule being a premise with condition(s)). Your task is to evaluate each rule in the list and select those that meet *any* of the following requirements:
- Full Condition Match: The fact fully and directly satisfies all condition(s) of the rule, allowing a valid derivation to obtain a new proposition.
- Partial Condition Match: The fact directly satisfies some, but not all, conditions of the rule. This means that additional fact(s) would be required to make a full derivation and obtain a new proposition.
If no rule is selected, return **None**.

Demonstrations:
`Example_1`
The given fact:
Bob is red.

The given list of rules:
All red, round people are quiet.
Red people are young.
If someone is round and smart then they are not red.
All white people are red.
Quiet people are green.
If someone is red and not white then they are not green.
If someone likes the dog and they are red then they are blue.

Let's go through each rule from the given list of rules and think step by step.
The selected rules (partial or full condition directly matched) are: *# Expected results*
All red, round people are quiet.
Red people are young.
If someone is red and not white then they are not green.
If someone likes the dog and they are red then they are blue.

`Example_2`
The given fact:
Anne is quiet.

The given list of rules:
If something is furry and not blue then it is nice.
If Anne is furry then Anne is nice.
Smart, furry things are round.

Let's go through each rule from the given list of rules and think step by step.
The selected rules (partial or full condition directly matched) are: *# Expected results*
None

Query:
The given fact:
`query_given_fact`

The given list of rules:
`query_given_list_of_rules`

Let's go through each rule from the given list of rules and think step by step.
The selected rules (partial or full condition directly matched) are:
`LLM_output`

Figure 18: The prompts for Forward Selection Module for *rule selection* in LogicTree.

Prompts for Backward Selection Module

Instructions:
Suppose you are one of the greatest AI scientists, logicians. Given a specific fact, a rule, and a repository of facts, your task is to identify the missing fact(s) required to fully satisfy the rule's conditions and check if the missing fact(s) exist in the fact repository.
- The given one specific fact already satisfies one of the rule's conditions. Identify the missing fact(s) needed to fully satisfy the rule.
- Automatically adapt pronouns (e.g., 'they', 'something', 'someone') to the correct subject based on the context of the given rule and the given fact.
- Check if the missing fact(s) are present in the fact repository.
  - If the missing fact(s) are present in the fact repository, return **True** along with the identified missing fact(s).
  - Otherwise, return **False**.

Demonstrations:
`Example_1`
The given one specific fact:
The cat likes the rabbit.

The given rule:
If someone is cold and they like the rabbit then the rabbit likes the cat.

The given fact repository:
The cat eats the bear.
The cat is cold.
The cat is kind.
The cat likes the rabbit.
The rabbit likes the tiger.
The tiger likes the bear.
The tiger visits the cat.

Let's go through each condition of the given rule. First identify the missing fact(s) needed to fully satisfy the rule. Then check if the missing fact(s) are present in the fact repository: *# Expected results*
The cat is cold.
True. The identified missing fact(s) in the fact repository: The cat is cold.

`Example_2`
The given one specific fact:
The rabbit likes the squirrel.

The given rule:
If someone likes the squirrel and the squirrel sees the cow then they are red.

The given fact repository:
The cow likes the rabbit.
The cow needs the mouse.
The mouse likes the squirrel.
The rabbit needs the cow.
The rabbit sees the cow.
The squirrel is nice.
The squirrel needs the cow.
The rabbit likes the squirrel.

Let's go through each condition of the given rule. First identify the missing fact(s) needed to fully satisfy the rule. Then check if the missing fact(s) are present in the fact repository: *# Expected results*
The squirrel sees the cow.
False

Query:
The given one specific fact:
`query_given_fact`

The given rule:
`query_given_rule`

The given fact repository:
`query_given_fact_repo`

Let's go through each condition of the given rule. First identify the missing fact(s) needed to fully satisfy the rule. Then check if the missing fact(s) are present in the fact repository:
`LLM_output`

Figure 19: The prompts for Backward Selection Module for *fact selection* in LogicTree.

Prompts for Derivation Module

Instructions:
Suppose you are one of the greatest AI scientists, logicians. Your task is to derive a new **Proposition** based on a given **rule** and some **fact(s)**.
Follow these instructions carefully:
1. Ensure that the **Proposition**:
   - Must be a valid logical derivation from the provided **rule** and **fact(s)**.
   - Must not duplicate any of the provided **fact(s)**.
   - Must not include any information not directly derived from the provided information.
   - Automatically adapt pronouns (e.g., 'they', 'something', 'someone') to the correct subject based on the context.
2. Do not apply the rule unless all conditions of the rule are met.
3. If no new **Proposition** can be derived, return **None**, and classify the reason into one of the following categories:
   - A. **Partial Information Met**: The given fact(s) meet some but not all conditions of the given rule.
   - B. **No Information Met**: The given fact(s) do not meet any conditions of the given rule.

Demonstrations:
`Example_1`
The given fact(s):
Erin is tall. Erin is cold.

The given rule:
Cold, tall people are not furry.

The derived proposition is: *# Expected results*
Erin is not furry.

`Example_2`
The given fact(s):
Bob is round.

The given rule:
If someone is round and smart then they are not red.

The derived proposition is: *# Expected results (pseudo dead-end)*
None
Reason: A. **Partial Information Met**: The given fact(s) meet some but not all conditions of the given rule.

`Example_3`
The given fact(s):
Alice is happy.

The given rule:
If Alice is sad and red, she is quiet.

The derived proposition is: *# Expected results (dead end)*
None
Reason: B. **No Information Met**: The given fact(s) do not meet any conditions of the given rule.

Query:
The given fact(s):
`query_given_facts`

The given rule:
`query_given_rule`

The derived proposition is:
`LLM_output`

Figure 20: The prompts for Derivation Module in LogicTree.

Prompts for Verification Module

Instructions:
Suppose you are one of the greatest AI scientists, logicians. Your task is to verify the relationship between a given **Proposition** and a **Conclusion**. There are three possibilities:
1. **Same:** The **Proposition** is directly equivalent to the **Conclusion**, meaning both the subject and the predicate (attributes) are the same.
2. **Opposite:** The **Proposition** directly contradicts the **Conclusion**. The subjects are the same, but the predicates (attributes) are in direct opposition, such as 'predicate' versus 'not predicate'.
3. **Indeterminate:** Neither **Same** nor **Opposite**. The **Proposition** and the **Conclusion** either have different predicates (attributes) or there is no clear relationship between them.

Demonstrations:
`Example_1`
Proposition:
Erin is not round.

Conclusion:
Erin is not green.

Verify the relationship between the given Proposition and the Conclusion: *# Expected results*
Indeterminate

`Example_2`
Proposition:
The rabbit is cold.

Conclusion:
The rabbit is cold.

Verify the relationship between the given Proposition and the Conclusion: *# Expected results*
Same

`Example_3`
Proposition:
The tiger is not young.

Conclusion:
The tiger is young.

Verify the relationship between the given Proposition and the Conclusion: *# Expected results*
Opposite

Query:
Proposition:
`query_proposition`

Conclusion:
`query_conclusion`

Verify the relationship between the given Proposition and the Conclusion:
`LLM_output`

Figure 21: The prompts for Verification Module in LogicTree.

Prompts for Fact Ranking Module (ablation study)

Instructions:
Imagine you are one of the greatest AI scientists, logicians. You are given a logic reasoning question that involves: a list of facts, a list of rules, a hypothesis to be verified.
Your task is to plan and prioritize the reasoning path:
- Sort the given **facts** based on their likelihood of being the starting point in the correct reasoning path to verify the hypothesis.
- The first fact in the sorted list should have the highest probability of being the right starting point, and the last fact should have the lowest probability.

Demonstrations:
`Example_1`
The given list of facts:
Bob is young.
Dave is blue.
Erin is blue.
Fiona is blue.
Fiona is kind.
Fiona is quiet.
Fiona is white.

The given list of rules:
If someone is kind then they are white.
Young people are quiet.
If someone is kind and white then they are blue.
All quiet, kind people are white.
If someone is quiet then they are kind.
If someone is white then they are young.
All blue, kind people are green.

The hypothesis to be verified:
Fiona is not green.

Let's sort the given **facts** based on their likelihood of being the starting point in the correct reasoning path.
The sorted facts are (each fact in a new line): *# Expected results*
Fiona is blue.
Fiona is kind.
Fiona is quiet.
Fiona is white.
Bob is young.
Dave is blue.
Erin is blue.

`Example_2`
.....

Query:
The given list of facts:
`query_fact_list`

The given list of rules:
`query_rule_list`

The hypothesis to be verified:
`query_hypothesis`

Let's sort the given **facts** based on their likelihood of being the starting point in the correct reasoning path.
The sorted facts are (each fact in a new line):
`LLM_output`

Figure 22: Prompts for Fact Ranking Module, used in the ablation study reported in Table 2.

Prompts for Rule Ranking Module (ablation study)

Instructions:
Imagine you are one of the greatest AI scientists, logicians. You are given a logic reasoning question that involves: a list of facts, a list of rules, a hypothesis to be verified.
Additionally, you are provided with a set of **selected rules**, which serve as potential intermediate steps in the reasoning process.
Your task is to plan and prioritize the reasoning path:
- Sort the **selected rules** based on their likelihood of being part of the correct reasoning path.
- The first rule in the sorted list should have the highest probability of being in the correct reasoning path, and the last rule should have the lowest probability.

Demonstrations:
`Example_1`
The given list of facts:
Bob is young.
Dave is blue.
Erin is blue.
Fiona is blue.
Fiona is kind.
Fiona is quiet.
Fiona is white.

The given list of rules:
If someone is kind then they are white.
Young people are quiet.
If someone is kind and white then they are blue.
All quiet, kind people are white.
If someone is quiet then they are kind.
If someone is white then they are young.
All blue, kind people are green.

The hypothesis to be verified:
Fiona is not green.

The given set of **selected rules**:
If someone is kind then they are white.
If someone is kind and white then they are blue.
All quiet, kind people are white.
All blue, kind people are green.

Let's sort the given **selected rules** based on their likelihood of being part of the correct reasoning path.
The sorted rules are (each rule in a new line): *# Expected results*
All blue, kind people are green.
If someone is kind then they are white.
If someone is kind and white then they are blue.
All quiet, kind people are white.

`Example_2`
.....

Query:
The given list of facts:
`query_fact_list`

The given list of rules:
`query_rule_list`

The hypothesis to be verified:
`query_hypothesis`

The given set of **selected rules**:
`query_selected_rules`

Let's sort the given **selected rules** based on their likelihood of being part of the correct reasoning path.
The sorted rules are (each rule in a new line):
`LLM_output`

Figure 23: Prompts for Rule Ranking Module, used in the ablation study reported in Table 2.