Dissecting Logical Reasoning in LLMs: A Fine-Grained Evaluation and Supervision Study

Anonymous ACL submission

Abstract

Logical reasoning is a core capability for many applications of large language models (LLMs), yet existing benchmarks often rely solely on final-answer accuracy, failing to capture the quality and structure of the reasoning process. We propose FineLogic, a fine-grained evaluation framework that assesses logical reasoning across three dimensions: overall benchmark accuracy, stepwise soundness, and representation-level alignment. In addition, to better understand how reasoning capabilities emerge, we conduct a comprehensive study on the effects of supervision format during fine-tuning. We construct four supervision styles (one natural language and three symbolic variants) and train LLMs under each. Our findings reveal that natural language supervision yields strong generalization even on out-of-distribution and long-context tasks, while symbolic reasoning styles promote more structurally sound and atomic inference chains. Further, our representation-level probing shows that fine-tuning primarily improves reasoning behaviors through step-by-step generation, rather than enhancing shortcut prediction or internalized correctness. Together, our framework and analysis provide a more rigorous and interpretable lens for evaluating and improving logical reasoning in LLMs. The code is available at https://anonymous.4open.science/r/FineLogic.

1 Introduction

Large language models (LLMs) are rapidly emerging as transformative tools across a wide array of applications (Achiam et al., 2023; Guo et al., 2024b; Thirunavukarasu et al., 2023; Nam et al., 2024). Among these, reasoning serves as a core capability underpinning tasks such as problem-solving (Lu et al., 2023), scientific question answering (Guo et al., 2024a), and code analysis (Nam et al., 2024). Consequently, a growing body of research has sought to evaluate and enhance the reasoning abilities of LLMs from multiple perspectives (Wei et al., 2022; Guo et al., 2025, 2024a). Within this broader landscape, logical reasoning stands out as a particularly challenging and intellectually demanding domain (Saparov and He, 2022a). It requires a synthesis of natural language understanding, formal logical interpretation, and multi-step inferential processing (Patel et al., 2024; Saparov et al., 2023; Morishita et al., 2024).

Despite growing interest in the logical reasoning capabilities of LLMs, most existing benchmarks focus narrowly on whether a model produces the correct final answer (Patel et al., 2024; Parmar et al., 2024; Han et al., 2022). This binary evaluation, typically assessing only the correctness of a "True" or "False" output, can be misleading, as it fails to determine whether the model arrived at the answer through valid multi-step reasoning (Saparov and He, 2022a). Consequently, correct answers may reflect guesswork rather than genuine logical inference. We are thus motivated to address **RQ1: How to rigorously evaluate LLMs' step-by-step correctness in logical reasoning tasks, beyond the binary evaluation of the final answer?**

In parallel with benchmarking efforts, numerous methods have been proposed to enhance the multi-step logical reasoning abilities of LLMs. While many leverage inference-time strategies (Wang et al., 2025), in-context learning (Creswell et al., 2022; Xu et al., 2024), or external logical verifiers (Pan et al., 2023) to guide the model toward more rigorous reasoning, some recent studies explored supervised fine-tuning (SFT) as a more direct approach to enhancing logical reasoning (Morishita et al., 2024; Feng et al., 2023). For example, Morishita et al. (2024) proposes a synthetic logic corpus designed to offer broad and systematic coverage of logical knowledge. However, it remains unclear for this important question, **RQ2: What style of training data, natural language or formal logical**

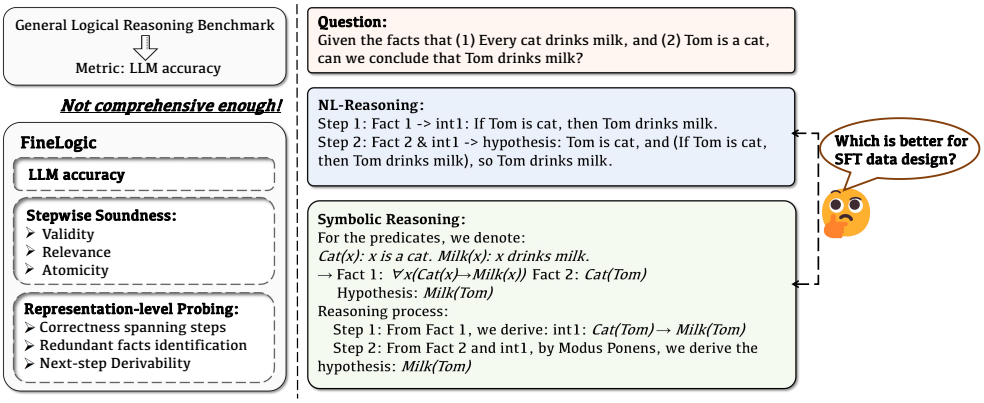


Figure 1: (Left) LLM logical reasoning evaluation: the general benchmark v.s. our fine-grained benchmark **FineLogic**. (Right) processing a logical reasoning task using natural language v.s. using symbolic methods.

symbols, better facilitates the learning of multi-step logical reasoning through SFT? Addressing this research question is important for understanding how to most effectively instill logical reasoning capabilities in LLMs.

To address RQ1, we propose **FineLogic**, a new evaluation framework designed to more fine-grainedly assess the logical reasoning capabilities of LLMs. Specifically, our framework evaluates models along three complementary dimensions: (1) **Overall benchmark accuracy**: This metric captures a model's ability to perform multi-step logical reasoning and its generalizability across problems from diverse domains. (2) **Stepwise Soundness**: Inspired by Saparov and He (2022a), we assess the quality of each intermediate reasoning step using three criteria—**validity** (whether the step is logically valid), **relevance** (whether its conclusion is used in later steps), and **atomicity** (whether it applies a single, minimal inference rule). These metrics aim to evaluate the model's ability to generate human-interpretable and logically coherent reasoning chains. (3) **Representation-level probing** (Ye et al., 2024): By applying probing techniques to LLM hidden representations, this evaluation provides insight into whether the model's understanding of logical structure is merely surface-level or embedded in its internal state.

To address RQ2, we systematically investigate how different supervision formats affect the reasoning capabilities of LLMs. Specifically, we examine both natural language-based training data and logic-symbol-based representations, including several structured variants. Our analysis shows that **natural language supervision is particularly effective in conveying core reasoning patterns**, leading to strong performance across a wide range of evaluation benchmarks. Notably, it exhibits impressive **generalizability even on out-of-distribution test sets** that require long reasoning chains. However, a deeper examination of *stepwise soundness* and *internal representation probing* reveals certain limitations. Models trained with natural language supervision tend to struggle with producing strictly minimal reasoning chains (e.g., more likely including **redundant steps** and applying multiple inference rules in a single step, as shown in Figure 5). In contrast, models trained with symbolic reasoning styles are better at filtering out irrelevant information, generating atomic steps aligned with individual deduction rules, and maintaining cleaner, logically grounded reasoning trajectories.

To summarize, our contributions are as follows:

- We propose FineLogic, a unified and rigorous evaluation framework for assessing LLMs' logical reasoning, moving beyond final-answer accuracy to evaluate the quality, interpretability, and coherence of their solutions.
- We conduct a comprehensive study on the effects of supervision format, fine-tuning LLMs on both natural language and symbolic logic data to examine their impact on reasoning across general and complex tasks.
- Through systematic analysis of models trained with different supervision styles, we identify key trade-offs between generalization and structural reasoning quality. These findings provide concrete insights into the design and selection of effective training data for supervised logical rea-

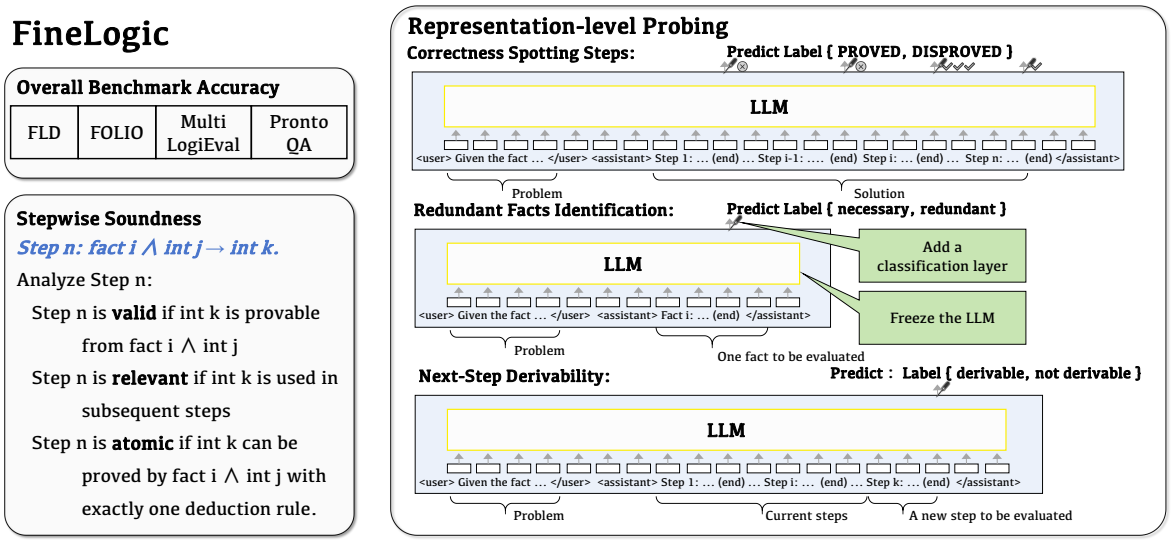


Figure 2: Overview of FineLogic, where overall benchmark accuracy, stepwise soundness, and representation-level probing are combined for a fine-grained evaluation of LLM's logical reasoning ability.

soning.

2 Related Works

Logical Reasoning Benchmarks. Numerous benchmarks have been proposed to evaluate the logical reasoning abilities of LLMs. Liu et al. (2023); Luo et al. (2023); Havrilla et al. (2024) mix logical and commonsense reasoning, making it hard to isolate logical competence. Others assess multi-step reasoning but rely only on final-answer accuracy (Parmar et al., 2024; Han et al., 2022; Tafjord et al., 2020; Mondorf and Plank, 2024). While ProntoQA (Saparov and He, 2022a; Saparov et al., 2023) introduces stepwise evaluation, it uses short problems and focuses only on step correctness. In contrast, our FineLogic framework provides a more rigorous and comprehensive assessment across sample-level correctness, step-level reasoning quality, and internal representation alignment.

Logical Reasoning Enhancement. Several studies have aimed to improve LLMs' performance on logical reasoning tasks. Some approaches rely on translating inputs into formal logic and using programmable verifiers to solve problems (Olausson et al., 2023; Pan et al., 2023; Yang et al., 2023; Ryu et al., 2024), which bypasses the model's own reasoning process. Others use in-context learning or inference-time strategies to guide output without fundamentally enhancing reasoning ability (Creswell et al., 2022; Wang et al., 2025; Xu et al., 2024; Sun et al., 2023; Toroghi et al., 2024). While a few works have explored fine-tuning or reinforcement learning to strengthen logical reasoning (Feng et al., 2023; Morishita et al., 2023, 2024; Xie et al., 2025; Yang et al., 2022; Xie et al., 2024), they have not examined which types of supervision are most effective for teaching LLMs to reason. In this work, we focus specifically on this open question.

3 FineLogic Evaluation Framework

As illustrated in Figure 2, FineLogic builds on existing benchmarks and evaluates logical reasoning ability from three complementary perspectives: (1) **Overall benchmark accuracy**, which measures whether the model can correctly solve multi-step reasoning tasks; (2) **Stepwise soundness**, which evaluates whether each reasoning step is valid and interpretable; (3) **Representation-level probing**, which assesses whether the model internally captures the problem's reasoning structure beyond surface-level patterns.

3.1 Overall Benchmark Accuracy

Similar to most benchmarks, our overall benchmark accuracy focuses on final-answer correctness. While coarse-grained, it offers a quick and effective way to assess a model's overall reasoning ability and cross-domain generalization. We evaluate on four challenging multi-step reasoning benchmarks: FLD (Morishita et al., 2024), FOLIO (Han et al., 2022), Multi-LogiEval (Patel et al., 2024), and ProntoQA (Saparov and He, 2022a). For FLD, we generate 50 samples per step (0–19) and 100 UNKNOWN cases. For FOLIO, the full test set is

Dataset	Samples	Label Types
FLD (Morishita et al., 2024)	1100	{T, F, Unknown}
FOLIO (Han et al., 2022)	203	{T, F, Unknown}
Multi-Logical (Patel et al., 2024)	390	{T, F}
Pronto-QA (Saparov and He, 2022a)	500	{T, F}

Table 1: Sample counts and label types for each dataset.

used. For Multi-LogiEval, we select first-order and propositional problems with depths 4–5. For ProntoQA, we follow Pan et al. (2023) and evaluate on the 500 hardest 5-hop samples. Dataset statistics are shown in Table 1, with details in Appendix A.1.

3.2 Stepwise Soundness

Building on Saparov and He (2022a), we evaluate the soundness of each intermediate reasoning step along three dimensions: **validity** (whether the step logically follows from its premises), **relevance** (whether its conclusion is used in later steps), and **atomicity** (whether it applies a single, minimal inference rule).

To assess these criteria, we extract the premises and conclusion of each step. We use GPT-4.1-mini to evaluate *validity* and *atomicity*. Manual verification on 200 annotated steps shows that GPT-4.1-mini achieves over 98% accuracy on both metrics. For *relevance*, we determine whether the conclusion of step i (e.g., `int` j) is referenced in any subsequent step $k > i$.

We then compute the proportion of samples in which *all* steps are valid, relevant, and atomic, providing a sample-level measure of reasoning integrity. Full prompt templates are provided in Figures 13 and 14.

3.3 Representation-level Probing

Inspired by Ye et al. (2024), we introduce **representation-level probing accuracy** to assess whether LLMs internally understand how and when to perform specific reasoning step. Unlike behavioral metrics, this method aligns internal representations with reasoning structure and tracks how reasoning knowledge evolves across steps.

We construct probing datasets from FLD test samples requiring 10–20 reasoning steps, using 450 problems for training and 100 for testing across three tasks, implementation details are provided in Appendix B:

Correctness Spanning Steps (CSS): Identifies the earliest step after which the model consistently predicts the correct label. The spanning length is the number of remaining steps from that point to the end. Higher accuracy indicates earlier internalization of the correct answer.

Redundant Facts Identification (RFI): After presenting all facts and the hypothesis, we append three necessary and three redundant facts. A classifier is trained to distinguish between them, measuring the model's ability to identify irrelevant information. Higher accuracy reflects better fact discrimination.

Next-Step Derivability (NSD): At six randomly selected intermediate steps, we append three valid and three invalid candidate steps. Probing predicts which are currently derivable. Higher accuracy indicates stronger awareness of valid next steps.

Our evaluation builds on two prior lines of work—stepwise reasoning evaluation (Saparov and He, 2022a) and representation-level probing (Ye et al., 2024)—but introduces key extensions tailored to logical reasoning.

Stepwise Soundness Evaluation. Saparov and He (2022a) evaluate reasoning steps using three criteria: **validity** (logical entailment), **utility** (contribution to the final proof), and **atomicity** (single rule application per step). Since *utility* depends on gold proof annotations and is often impractical, we propose a more accessible alternative: **relevance**, which checks whether a step's conclusion is used in any subsequent inference. Moreover, prior work focuses on individual steps, while we extend this to the solution level by introducing **all-steps validity, relevance, and atomicity**—sample-level metrics that reflect whether a full reasoning chain is logically sound and interpretable.

Representation-Level Probing. Ye et al. (2024) use probing to assess internal reasoning in math problems. We adapt this method to logical reasoning and introduce a new metric: **Correctness Spanning Steps (CSS)**, which identifies the earliest point after which the model consistently predicts the correct label. CSS approximates the model's internal reasoning depth by measuring how early it stabilizes on the correct answer.

4 Supervision Format and Style: SFT Data Design

In this section, we examine how different supervision styles for SFT affect the logical reasoning abilities of LLMs. Our training data is based on

FLD and **ProntoQA**, both of which include gold reasoning chains suitable for constructing diverse supervision styles.

For **FLD**, we generate 500 problems for each reasoning depth from 0 to 15, plus 1500 UNKNOWN samples, totaling 9500 training instances. For **ProntoQA**, we use 3200 3-hop problems. During evaluation, FLD covers depths 0–19, while ProntoQA uses only the hardest 5-hop samples.

We compare four supervision styles across two categories: natural language-based and symbolic reasoning. Each style reflects a different level of abstraction and clarity in reasoning structure.

- **NL-Reasoning**: Solutions are written entirely in natural language, with no intermediate symbolization or abstraction.
- **Symbolic Reasoning (Structured)**: Problems are formalized by defining variables and predicates, translating facts and hypotheses into logical forms, and reasoning step by step using symbolic logic.
- **Symbolic Reasoning (Filtered)**: A simplified variant where only necessary facts are retained, shortening reasoning chains and reducing input complexity.
- **Symbolic Reasoning (Direct)**: Facts are directly expressed in symbolic form without defining variables or predicates, which shortens sequences but may introduce ambiguity.

A small portion of translations, connective phrases, and intermediate steps are generated using GPT-4.1. Prompt examples are shown in Figure 4 (Appendix E).

5 Experiments

5.1 Experimental Setup

We conduct all SFT experiments on two models: LLaMA-3.1-8B-Instruct and Qwen-2.5-7B-Instruct, both fully fine-tuned for 3 epochs with a learning rate of 1×10^{-6}.

Our baselines include four models: LLaMA-3.1, Qwen-2.5, GPT-4o, and DeepSeek R1. Fine-tuning-based methods use only LLaMA and Qwen as base models. Due to computational constraints, representation-level probing is conducted only on LLaMA, Qwen, and their SFT variants. Stepwise evaluation requires strict output formatting and enforces explicit step-by-step generation.

We compare SFT models trained with different supervision styles against these baselines:

- **Direct Answer**
- **Chain-of-Thought (CoT)** (Wei et al., 2022)
- **Few-Shot Learning** (Brown et al., 2020)
- **LOGIPT** (Creswell et al., 2022)
- **Selection-Inference** (Creswell et al., 2022)
- **SymbCoT** (Xu et al., 2024)
- **LogicLM** (Pan et al., 2023)

More detailed experimental setups can be found in Appendix A.

5.2 Results

We conducted experiments for analyzing the performance of four models combined with various prompting and fine-tuning settings under the **Fine-Logic Evaluation Framework**.

5.2.1 Results on Overall Benchmark Accuracy

As shown in Table 1, we report the **overall benchmark accuracy** across four datasets, as well as the **step-wise accuracy** on the FLD benchmark, stratified by reasoning depth (Figure 3. Our analysis yields several key observations:

CoT and few-shot prompting generally improve performance, but baseline methods do not consistently yield gains. Across the four evaluation datasets, both CoT and few-shot prompting lead to broadly positive improvements, indicating their general effectiveness in enhancing LLM performance on logical reasoning tasks. Notably, **few-shot prompting consistently outperforms CoT**, suggesting that for complex logical tasks, showing the model *how* to think (via exemplars) is more beneficial than simply encouraging it to reason step by step. This may be because logical questions naturally elicit multi-step reasoning under direct prompting, limiting the marginal benefit of CoT. In contrast, few-shot demonstrations provide clearer procedural scaffolding, which appears more effective in guiding the model's reasoning process.

In contrast, baseline prompting methods such as *Logic-LM*, *SymbCoT*, and *Sel-Inf* show inconsistent performance and sometimes underperform even direct prompting. For example, *Logic-LM* performs well on simpler problems but degrades on complex ones, with Qwen's Multi-LogiEval ac-

Model	Setting	FLD	FOLIO	Multi-LogiEval	ProntoQA
GPT-4o	Direct	53.0	72.4	71.0	98.8
	CoT	54.1	69.5	76.9	98.6
	Few-shot	**58.3**	**74.4**	84.4	99.0
	Logic-LM	46.9	72.1	83.3	**100**
	SymbCoT	47.6	71.6	72.1	**100**
	Sel-Inf	51.9	66.5	**84.9**	94.4
DeepSeek-R1	Direct	77.2	75.9	81.8	**100**
	CoT	77.6	78.8	79.0	**100**
	Few-shot	77.3	81.8	**84.6**	99.4
	Logic-LM	69.6	77.5	81.2	96.4
	SymbCoT	69.6	82.8	72.0	98.2
	Sel-Inf	**83.8**	**85.2**	73.1	96.0
Llama-3.1-8B-Instruct	Direct	31.7	54.7	40.5	64.6
	CoT	29.3	50.7	44.6	63.8
	Few-shot	41.0	46.5	59.4	48.9
	Logic-LM	38.3	52.5	44.4	77.6
	SymbCoT	38.1	58.8	46.3	78.8
	Sel-Inf	48.5	47.5	55.2	64.2
	LogiPT	53.3	**61.7**	57.9	76.4
	SFT-NL	**67.5**	57.1	**71.3**	99.6
	SFT-Symb-Struct	63.2	56.2	59.7	**99.8**
	SFT-Symb-Filter	66.7	54.7	50.8	91.0
	SFT-Symb-Direct	52.8	48.3	53.9	98.8
Qwen-2.5-7B-Instruct	Direct	46.6	61.1	37.0	90.6
	CoT	50.4	65.5	54.3	90.4
	Few-shot	53.2	68.5	61.3	91.1
	Logic-LM	46.6	**69.1**	27.1	85.8
	SymbCoT	22.6	57.5	**63.9**	87.0
	Sel-Inf	49.0	62.6	39.7	92.6
	LogiPT	58.6	61.7	55.6	52.4
	SFT-NL	**71.0**	62.6	64.3	**97.4**
	SFT-Symb-Struct	54.6	50.7	57.7	83.8
	SFT-Symb-Filter	54.7	55.7	61.0	96.0
	SFT-Symb-Direct	54.8	53.2	58.7	61.4

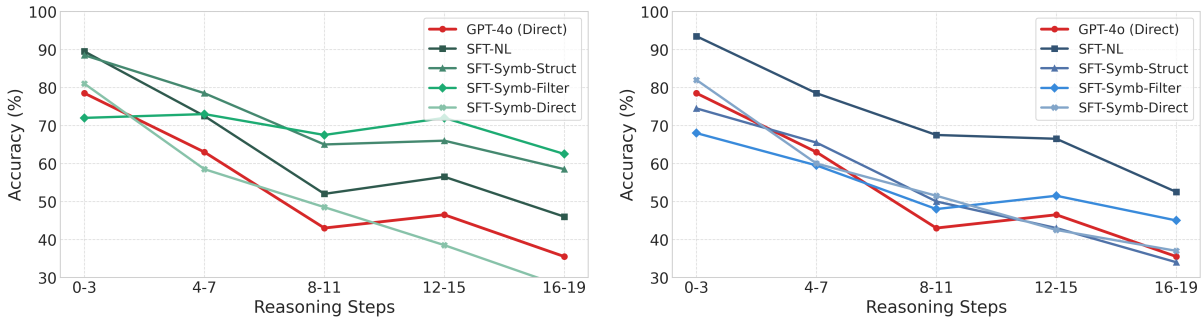
Table 2: Overall Benchmark Accuracy on four models with different settings.

curacy dropping to 27.1%. *SymbCoT* sometimes improves over *Logic-LM* (e.g., 63.8% on Multi-LogiEval with Qwen) but also shows large drops elsewhere (e.g., 22.6% on FLD, versus 44.6% with direct prompting).

Supervised fine-tuning outperforms inference-time methods, but its effectiveness heavily depends on the supervision style. Compared to inference-time prompting strategies, SFT yields significantly greater improvements in logical reasoning performance. Among all training styles, **natural language-based supervision (SFT-NL) produces the most substantial and consistent gains across datasets and models.**

Notably, even though SFT was conducted using only problems from FLD and ProntoQA with reasoning depths *less than those in the test set*, the resulting models show robust improvements. For example, under the SFT-NL setting, Llama's accuracy on FLD increased from 31.7% (direct prompting) to 67.5% and Qwen improved from 46.6% to 71.0%, approaching the best-performing baseline DeepSeek R1. On ProntoQA, most SFT variants achieve over 90% accuracy. Furthermore, even on out-of-distribution datasets such as FOLIO and Multi-LogiEval, some SFT settings deliver strong generalization. For instance, on Multi-LogiEval, Llama with SFT-NL improved to 71.3%, matching the performance of GPT-4o.

While SFT-NL demonstrates the best overall and most transferable performance, other styles of supervision yield much smaller gains. This may be since LLMs are primarily pretrained on natural language data, making symbolic reasoning—especially when it requires both translation and inference over logic forms—significantly more challenging. Among the symbolic settings, **SFT-Symb-Filter** consistently outperforms other vari-



(a) Performance of Llama-3.1-8B-Instruct SFT.

(b) Performance of Qwen-2.5-7B-Instruct SFT.

Figure 3: Comparison of SFT variants' performance across different reasoning step ranges in FLD dataset. Both charts show accuracy declines with increasing inference steps, with GPT-4o (Direct) included as a reference. In (a), Llama with SFT-Symb-Filter maintains strong performance even in the 16-19 step range (out-of-distribution), while in (b), Qwen with SFT-NL shows remarkable early-stage reasoning capabilities.

ants. By removing redundant reasoning steps from the symbolic training data, this setting simplifies training and enhances performance. In contrast, **SFT-Symb-Direct**, which skips variable and predicate definitions entirely, performs poorly, likely due to the introduction of ambiguity and the lack of explicit logical structure.

Accuracy declines with reasoning depth, but SFT enables small models to match GPT-4o even on the most challenging out-of-distribution samples. As shown in Figure 3, model accuracy decreases as the required number of reasoning steps increases. Nonetheless, our results show that SFT substantially improves model robustness, even on long-chain, out-of-distribution examples. On in-distribution FLD test problems (0–15 steps), SFT models trained under most styles outperform GPT-4o. For instance, across reasoning depths up to 15, both Llama and Qwen with SFT-NL surpass GPT-4o's performance.

On more difficult out-of-distribution questions requiring 16–19 steps of reasoning—where no training samples are available—performance drops by approximately 10% relative to the 12–15 step range. However, even under these conditions, SFT models maintain accuracy comparable to GPT-4o. Combined with strong generalization to unseen datasets such as FOLIO and Multi-LogiEval, these results suggest that **SFT induces genuine logical reasoning ability in LLMs**. At the same time, the sharp performance decline on longer reasoning chains implies that some portion of success on shorter problems may still stem from shallow pattern matching or memorization, rather than robust inference. Detailed results can be found in C.

Model	Setting	All Valid	All Relevant	All Atomic
GPT-4o	Few-shot	7.6	**56.2**	4.4
Deepseek-R1	Few-shot	13.1	33.8	5.7
Llama-3.1-8B-Instruct	Few-shot	4.5	17.4	1.6
	LogiPT	5.2	**28.5**	4.9
	SFT-NL	**40.9**	8.5	13.0
	SFT-Symb-Struct	35.0	15.4	24.7
	SFT-Symb-Filter	21.8	16.9	12.4
	SFT-Symb-Direct	33.7	10.2	**25.1**
Qwen-2.5-7B-Instruct	Few-shot	10.1	35.1	2.6
	LogiPT	6.4	**39.8**	5.3
	SFT-NL	27.6	5.4	8.5
	SFT-Symb-Struct	**35.3**	9.1	**19.8**
	SFT-Symb-Filter	16.7	11.7	10.5
	SFT-Symb-Direct	19.7	0.3	11.9

Table 3: Stepwise soundness of various models under settings without inference-time interventions. The best variant of Llama and Qwen is highlighted.

5.2.2 Results on Stepwise Soundness

Table 3 reports the results of **stepwise soundness evaluation** across different models and training settings, offering a more fine-grained view of how well LLMs internalize logical reasoning principles.

The **All Valid** metric measures the proportion of samples in which *every* generated reasoning step is logically valid. This is a stringent indicator of a model's grasp of formal reasoning rules. We observe that models trained with the **SFT-NL** and **SFT-Symb-Struct** settings achieve particularly high All Valid scores—substantially outperforming even GPT-4o and DeepSeek-R1. Notably, the Llama model fine-tuned under SFT-NL achieves an All Valid rate of 40.9

The **All Relevant** metric measures the proportion of samples in which every generated step is *relevant*—i.e., none of the steps are redundant or un-

Model	Setting	CSS	RFI	NSD
Llama-3.1-8B-Instruct	–	8.0	9.9	32.0
	LogiPT	8.1	0.7	44.2
	SFT-NL	8.5	9.9	**51.5**
	SFT-Symb-Struct	8.7	11.1	36.1
	SFT-Symb-Filter	**9.7**	11.1	46.4
	SFT-Symb-Direct	9.0	**18.5**	41.2
Qwen-2.5-7B-Instruct	–	**8.6**	7.4	43.3
	LogiPT	8.1	9.2	43.2
	SFT-NL	8.2	16.0	44.3
	SFT-Symb-Struct	8.5	14.8	43.3
	SFT-Symb-Filter	8.3	16.0	**45.4**
	SFT-Symb-Direct	8.6	**18.5**	43.3

Table 4: Evaluation of Correctness Spanning Steps (CSS), Redundant Fact Identification (RFI), and Next-step Derivability (NSD) on Llama and Qwen. '-' indicates the original model. The best variant is highlighted.

necessary for reaching the conclusion. GPT-4o and LogiPT perform exceptionally well on this metric, implying that they rarely generate superfluous reasoning steps. In contrast, SFT-NL and SFT-Symb-Direct consistently underperform. For SFT-NL, this may stem from the nature of natural language reasoning: due to its semantic richness and lack of structural constraints, the model may occasionally include exploratory or overly verbose steps, unsure of which inference is most effective. For SFT-Symb-Direct, the poor performance is likely due to the model may failure to fully capture inter-fact dependencies, resulting in reasoning sequences that are logically valid but contain unused or irrelevant steps.

The **All Atomic** metric evaluates whether every step in a reasoning chain corresponds to a single atomic inference—i.e., whether steps avoid combining multiple logical moves. Here, **SFT-Symb-Struct** consistently outperforms other settings, highlighting the advantages of structured symbolic reasoning. Symbolic reasoning is inherently more compact and constrained, which likely helps the model learn what constitutes a minimal, rule-aligned inference step. In contrast, natural language reasoning often fuses multiple reasoning rules into a single step, making it harder for the model to isolate atomic operations.

5.2.3 Results on Representation-level Probing

Table 4 presents results from our **representation-level probing analysis**, which aims to assess whether the models have internally acquired key reasoning abilities.

Regarding **Correctness Spanning Steps (CSS)**, which assesses how early the model predicts the final answer, most SFT methods show little improvement. Only **SFT-Symb-Filter** on Llama yields a modest gain, suggesting SFT primarily guides step-by-step generation rather than enhancing early "shortcut" predictions.

For the **Redundant Fact Identification** metric, most SFT settings show noticeable gains. Interestingly, **SFT-Symb-Direct** consistently achieves the highest performance. We hypothesize that this setting, which omits the explicit logic translation phase, forces the model to implicitly learn both symbolic interpretation and reasoning. In doing so, it may develop a stronger understanding of the logical roles and dependencies among the facts, thus improving its ability to distinguish between relevant and redundant conditions.

In the **Next-Step Derivability** task, SFT consistently benefits Llama, particularly **SFT-NL** (likely due to natural language's accessibility). In contrast, SFT shows minimal impact on Qwen, possibly because its base model is already proficient in step-tracking.

6 Conclusion

We introduce FineLogic, a unified and fine-grained framework for evaluating the logical reasoning capabilities of large language models. By integrating overall benchmark accuracy, stepwise soundness, and representation-level probing, FineLogic enables more interpretable and rigorous assessment beyond final-answer correctness. Leveraging this framework, we conduct a systematic investigation of how different fine-tuning supervision formats impact reasoning ability. Our experiments demonstrate that while natural language supervision leads to strong generalization and benchmark gains, symbolic styles better support minimal, rule-aligned reasoning structures. Furthermore, representation-level probing reveals that SFT primarily affects how models generate stepwise solutions rather than their ability to predict answers directly. These findings offer practical guidance for designing supervision strategies tailored to different reasoning objectives and highlight the importance of evaluating both behavioral and internal reasoning quality when advancing LLM reasoning systems.

Limitations

Our work has two main limitations. First, although FineLogic evaluates models across three complementary dimensions, it is built upon a limited set of datasets. While we selected benchmarks that span multiple domains and reasoning depths, no fixed dataset collection can offer a complete assessment of LLM reasoning capabilities. Nevertheless, the modular design of FineLogic allows it to be easily extended to future benchmarks, enabling the same set of evaluation tasks—overall accuracy, stepwise soundness, and representation-level probing—to be applied to new problem settings. Second, although we provide detailed analyses of how different supervision styles succeed or fail across tasks, we do not explore how to systematically integrate these insights into the design of more comprehensive or hybrid SFT datasets. We leave it to future work to develop adaptive or mixed-format training strategies that balance generalization, structural soundness, and interpretability in logical reasoning.

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*.

Placeholder Author and 1 others. 2025. Llm+al: Bridging large language models and action languages for complex reasoning about actions. In *AAAI*.

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.

Andy T. Chen and 1 others. 2023. Program of thoughts prompting: Disentangling computation from reasoning. In *ICLR*.

Wenhuchen Chen, Nafise Sadat Moosavi, and Mario Fritz. 2020. Logical natural language generation from open-domain tables. In *ACL*.

Antonia Creswell, Murray Shanahan, and Irina Higgins. 2022. Selection-inference: Exploiting large language models for interpretable logical reasoning. *arXiv preprint arXiv:2205.09712*.

Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, and 1 others. 2021. Explaining answers with entailment trees. In *EMNLP*.

Jiazhan Feng, Ruochen Xu, Junheng Hao, Hiteshi Sharma, Yelong Shen, Dongyan Zhao, and Weizhu Chen. 2023. Language models can be logical solvers. *arXiv preprint arXiv:2311.06158*.

Luyu Gao and 1 others. 2023. Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*.

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*.

Kehan Guo, Bozhao Nan, Yujun Zhou, Taicheng Guo, Zhichun Guo, Mihir Surve, Zhenwen Liang, Nitesh Chawla, Olaf Wiest, and Xiangliang Zhang. 2024a. Can llms solve molecule puzzles? a multimodal benchmark for molecular structure elucidation. *Advances in Neural Information Processing Systems*, 37:134721–134746.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. 2024b. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*.

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

Alex Havrilla, Sharath Raparthy, Christoforus Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, and Roberta Raileanu. 2024. Glore: When, where, and how to improve llm reasoning via global and local refinements. *arXiv preprint arXiv:2402.10963*.

Omar Khattab and 1 others. 2023. Dspy: Compiling declarative language-model calls into self-optimizing pipelines. In *ICLR*.

Jiawei Li and 1 others. 2023. Explicit planning helps language models in logical reasoning. In *ACL*.

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

Zheng Liu, Zixu Liu, Yiming Liu, and 1 others. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. In *IJCAI*.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. 2023. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. *arXiv preprint arXiv:2310.02255*.

Man Luo, Shrinidhi Kumbhar, Mihir Parmar, Neeraj Varshney, Pratyay Banerjee, Somak Aditya, Chitta Baral, and 1 others. 2023. Towards logiglue: A brief survey and a benchmark for analyzing logical reasoning capabilities of language models. *arXiv preprint*

arXiv:2310.00836.

Philipp Mondorf and Barbara Plank. 2024. Liar, Liar, Logical Mire: A Benchmark for Suppositional Reasoning in Large Language Models. *arXiv preprint*. ArXiv:2406.12546 [cs] TLDR: This paper introduces $\textit{TruthQuest}$, a benchmark for suppositional reasoning based on the principles of knights and knaves puzzles, and shows that large language models like Llama 3 and Mixtral-8x7B exhibit significant difficulties solving these tasks.

Terufumi Morishita, Gaku Morio, Atsuki Yamaguchi, and Yasuhiro Sogawa. 2023. Learning deductive reasoning from synthetic corpus based on formal logic. In *International Conference on Machine Learning*, pages 25254–25274. PMLR.

Terufumi Morishita, Gaku Morio, Atsuki Yamaguchi, and Yasuhiro Sogawa. 2024. Enhancing reasoning capabilities of llms via principled synthetic logic corpus. *Advances in Neural Information Processing Systems*, 37:73572–73604.

Daye Nam, Andrew Macvean, Vincent Hellendoorn, Bogdan Vasilescu, and Brad Myers. 2024. Using an llm to help with code understanding. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, pages 1–13.

Theo X Olausson, Alex Gu, Benjamin Lipkin, Cedegao E Zhang, Armando Solar-Lezama, Joshua B Tenenbaum, and Roger Levy. 2023. Linc: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. *arXiv preprint arXiv:2310.15164*.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. 2023. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. *arXiv preprint arXiv:2305.12295*.

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

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

Hyun Ryu, Gyeongman Kim, Hyemin S Lee, and Eunho Yang. 2024. Divide and translate: Compositional first-order logic translation and verification for complex logical reasoning. *arXiv preprint arXiv:2410.08047*.

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

Abulhair Saparov and He He. 2022b. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *arXiv:2210.01240*.

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

Koustuv Sinha, Chris Dyer, Dani Yogatama, and 1 others. 2019. Clutrr: A diagnostic benchmark for inductive reasoning from text. In *EMNLP*.

Hongda Sun, Weikai Xu, Wei Liu, Jian Luan, Bin Wang, Shuo Shang, Ji-Rong Wen, and Rui Yan. 2023. Determlr: Augmenting llm-based logical reasoning from indeterminacy to determinacy. *arXiv preprint arXiv:2310.18659*.

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

Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. 2023. Large language models in medicine. *Nature medicine*, 29(8):1930–1940.

Armin Toroghi, Willis Guo, Ali Pesaranghader, and Scott Sanner. 2024. Verifiable, debuggable, and repairable commonsense logical reasoning via llm-based theory resolution. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 6634–6652.

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

Xuezhi Wang and 1 others. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv:2203.11171*.

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.

Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. 2024. On memorization of large language models in logical reasoning. *arXiv preprint arXiv:2410.23123*.

Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. 2025. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*.

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

Kaiyu Yang, Jia Deng, and Danqi Chen. 2022. Generating natural language proofs with verifier-guided search. *arXiv preprint arXiv:2205.12443*.

Yuan Yang, Siheng Xiong, Ali Payani, Ehsan Shareghi,

and Faramarz Fekri. 2023. Harnessing the power of large language models for natural language to first-order logic translation. *arXiv preprint arXiv:2305.15541*.

Shunyu Yao and 1 others. 2022. React: Synergizing reasoning and acting in language models. *arXiv:2210.03629*.

Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. 2024. Physics of language models: Part 2.1, grade-school math and the hidden reasoning process. In *The Thirteenth International Conference on Learning Representations*.

Denny Zhou and 1 others. 2023. Least-to-most prompting enables complex reasoning in large language models. In *ICLR*.

A Detailed Experimental Setup

A.1 Detailed Dataset Information

In this section, we present the logical reasoning datasets used in our experiments. All datasets are publicly available. We describe the data sources and sampling procedures in detail below.

FLD (Morishita et al., 2024) The FLD dataset is constructed based on the number of reasoning steps (denoted as `steps`). We first generate 50,000 raw samples uniformly distributed across step depths from 0 to 15. For each step, we randomly sample 250 `PROVED` and 250 `DISPROVED` examples, yielding a total of $16 \times 500 = 8{,}000$ labeled samples. Additionally, we include 1,500 randomly drawn `UNKNOWN` instances, resulting in a total training set of 9,500 samples.

The test set follows a similar sampling process but spans a broader range of reasoning depths (0–19). We select 25 `PROVED` and 25 `DISPROVED` instances per step, yielding $20 \times 50 = 1{,}000$ labeled examples. We also sample 100 `UNKNOWN` instances, producing a test set of 1,100 samples(Sinha et al., 2019; Tafjord et al., 2020; Pan et al., 2023).

FOLIO (Han et al., 2022) The FOLIO dataset contains 203 test examples, all of which are included in our experiments(Chen et al., 2020; Liu et al., 2020).

Multi-LogiEval The Multi-LogiEval dataset consists of five subsets with different depth (denoted 'd1' through 'd5'). We include only the 'd4' and 'd5' subsets to maintain a consistent dataset size, yielding a total of 390 examples. From each, we extract all samples from the `First-Order Logic` and `Propositional Logic` categories, masking their logic type annotations to ensure a uniform setting(Dalvi et al., 2021).

ProntoQA We use the standard ProntoQA dataset rather than the OOD variant, following prior work (Pan et al., 2023). We use 500 hardest samples requiring 5-hop reasoning steps as a held-out test set. For training, we sample 3,200 problems that each require 3-hop reasoning, selected to provide sufficient multi-step depth while remaining tractable for supervised fine-tuning(Saparov and He, 2022b; Li et al., 2023).

A.2 Detailed Baseline Methods

LOGIPT (Feng et al., 2023) LOGIPT is a novel language model designed to emulate the reasoning processes of logical solvers. Unlike traditional solver-augmented models that parse natural language into symbolic representations before reasoning, LOGIPT directly learns to generate symbolic reasoning steps, bypassing potential parsing errors(Gao et al., 2023; Chen et al., 2023).

Selection-Inference (Creswell et al., 2022) Enhancing LLMs by n-shot learning between selecting relevant facts and inferring new information, enabling interpretable multi-step logical reasoning. In the experiments, we achieved rather low accuracy for limited questions (hypothesis) to trigger new inferences. Code is open-sourced(Yao et al., 2022; Wang et al., 2022; Zhou et al., 2023).

LogicLM (Pan et al., 2023) A neuro-symbolic framework that addresses LLMs' limitations in complex logical reasoning by integrating symbolic solvers with language models. The approach works through a three-stage pipeline: (1) using LLMs to translate natural language problems into symbolic formulations, (2) employing deterministic symbolic solvers to perform faithful logical inference, and (3) interpreting results back to natural language. It features a self-refinement module that iteratively revises symbolic representations based on solver error messages.

SymbCoT (Xu et al., 2024) The model introduces a novel framework that enhances LLMs by integrating symbolic expressions and logical rules into the CoT reasoning process. It translates natural language contexts into symbolic formats, derives step-by-step plans using symbolic logic, and employs a verifier to ensure the correctness of translations and reasoning chains(Author et al., 2025; Khattab et al., 2023; Gao et al., 2023).

B Representation-Level Probing Implementation Details

We design three probing tasks to assess whether the model's internal representations capture reasoning-relevant information during multi-step logical problem solving. All probing experiments are conducted on a subset of the **FLD** dataset, specifically the 550 most complex problems requiring 10–20 reasoning steps. We use 450 problems for training and 100 for evaluation.

B.1 Representation Extraction

For all probing tasks, we extract the hidden state of the **final token from the last transformer layer** after processing the input prefix. The prefix consists of all reasoning steps up to a target step k (i.e., steps 1 to k), and the final-token representation is treated as a summary of the model's internal reasoning state at that point.

B.2 Probing Model

We use a lightweight yet effective classifier to probe the information contained in these hidden states. Specifically, we adopt a **logistic regression classifier** with feature standardization and **5-fold cross-validation** for hyperparameter selection. This setup ensures a simple and interpretable linear decision boundary while maintaining robustness against overfitting. The classifier is trained solely on the extracted representations, while the underlying language model remains frozen throughout the probing process.

B.3 Task 1: Correctness Spanning Steps

This task evaluates how early in the reasoning process the model internalizes the correct final answer. For a problem requiring n reasoning steps, we:

- Generate n input prefixes, each ending at step i, where $i \in [1, n]$.
- Train a probing classifier to predict the ground-truth label (True / False) based on the representation at each prefix.
- For each test sample, we identify the smallest i such that the classifier correctly predicts the label at step i but fails at step $i-1$.

The **correctness spanning length** is defined as $n - i$, capturing how early the model "knows" the correct answer.

B.4 Task 2: Redundant Facts Identification

This task assesses whether the model can distinguish between relevant and irrelevant facts. For each sample:

- We locate the point after all facts and the hypothesis have been presented.
- We construct six variants of the input: three with **necessary facts** (used later in the proof), and three with **redundant facts** (unused in any proof step).

- The classifier is trained to predict whether the appended facts are necessary or redundant based on the updated representation.

This task tests whether the model encodes awareness of which premises are logically relevant for solving the task.

B.5 Task 3: Next-Step Derivability

This task probes whether the model can determine which steps are logically available at a given point in the proof. For each sample:

- We randomly select six intermediate steps.
- At each step, we append three **valid next steps** (that are inferable from the current context) and three **invalid steps** (that appear later in the proof but are not yet derivable).
- The classifier is trained to distinguish between currently valid and invalid steps.

This task evaluates whether the model has encoded an implicit understanding of the forward progression of logical inference.

C Experiment Details

This section provides further details on our experimental results. Table 5 presents a comprehensive breakdown of FLD accuracy across different reasoning step ranges for the evaluated models and settings. The data illustrates that while models fine-tuned with natural language supervision (e.g., Llama-3.1-SFT-NL achieving 89.5% accuracy for 0-3 steps on FLD) perform strongly on tasks with shallower reasoning depths, their symbolic reasoning counterparts tend to exhibit greater resilience as the complexity and number of reasoning steps increase. For instance, on FLD problems requiring 16-19 steps, Llama-3.1-SFT-Symb-Filter (62.5%) and Llama-3.1-SFT-Symb-Struct (58.5%) maintain higher accuracy compared to Llama-3.1-SFT-NL (46.0%), highlighting the benefit of symbolic formats for robust multi-step inference.

D Computational Resources

All supervised fine-tuning experiments were conducted using 4 NVIDIA A100 GPUs. Each model was trained for approximately 2 hours. Evaluation on the full suite of benchmarks and diagnostic metrics required an additional 0.5 hours per model.

E Example and Case Study

This section showcases examples from our training dataset along with an error case study. Further details can be found in Figure 4 and Figure 5.

F Prompt Template

This section showcases various prompts, encompassing those designed for reasoning and data generation, as detailed in Figures 6,7,8,9,10,11,12,13,14

Model	Setting	FLD Accuracy by Step				
		0–3	4–7	8–11	12–15	16–19
GPT-4o	Direct	78.5	63.0	43.0	46.5	35.5
	CoT	82.0	62.0	56.5	44.0	46.5
	Few-shot	81.5	68.5	53.5	46.0	47.0
	Logic-LM	68.4	52.1	31.5	28.2	22.8
	SymbCoT	69.9	52.5	32.0	26.5	24.5
	Sel-Inf	64.5	55.5	49.5	49.0	55.5
DeepSeek -R1	Direct	92.5	86.0	80.5	75.0	76.5
	CoT	92.0	86.0	78.0	77.5	73.5
	Few-shot	89.0	85.0	80.5	69.0	71.0
	Logic-LM	91.4	78.6	64.8	58.2	52.4
	SymbCoT	86.4	80.5	70.9	45.2	53.4
	Sel-Inf	93.0	88.0	84.5	79.0	75.5
Llama-3.1 -8B-Instruct	Direct	40.5	30.0	24.0	27.0	25.5
	CoT	41.5	32.0	29.0	24.0	19.5
	Few-shot	49.5	45.5	33.0	39.0	32.0
	Logic-LM	56.4	41.3	32.6	28.8	26.2
	SymbCoT	57.8	41.0	39.0	37.8	35.4
	Sel-Inf	63.5	55.5	52.5	45.0	42.0
	LogiPT	72.5	53.5	51.0	35.0	37.0
	SFT-NL	89.5	72.5	52.0	56.5	46.0
	SFT-Symb-Struct	88.5	78.5	65.0	66.0	58.5
	SFT-Symb-Filter	72.0	73.0	67.5	72.0	62.5
	SFT-Symb-Direct	81.0	58.5	48.5	38.5	27.5
Qwen-2.5 -7B-Instruct	Direct	69.0	45.5	45.0	38.5	36.0
	CoT	70.5	55.5	36.5	42.5	40.5
	Few-shot	63.0	44.0	33.5	27.0	33.0
	Logic-LM	68.7	51.2	31.4	26.0	29.2
	SymbCoT	52.3	39.5	30.7	28.1	19.9
	Sel-Inf	49.0	26.5	29.5	27.0	24.5
	LogiPT	80.5	74.0	64.0	68.0	57.5
	SFT-NL	93.5	78.5	67.5	66.5	52.5
	SFT-Symb-Struct	74.5	65.5	50.0	43.0	34.0
	SFT-Symb-Filter	68.0	59.5	48.0	51.5	45.0
	SFT-Symb-Direct	82.0	60.0	51.5	42.5	37.0

Table 5: FLD accuracy breakdown by reasoning step ranges

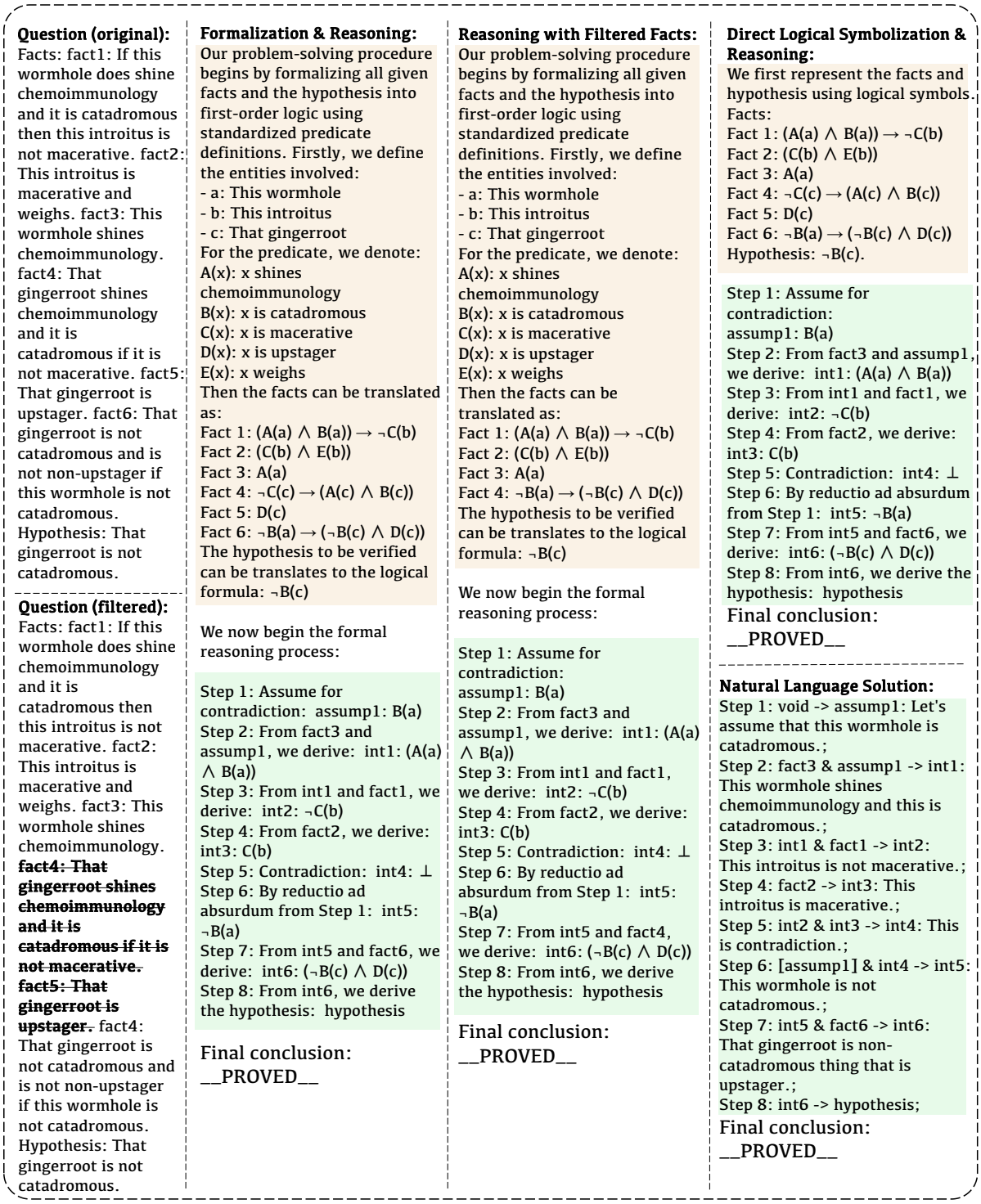


Figure 4: Comparison of a logical reasoning problem under four distinct training data settings. The figure illustrates: (a) direct logical symbolization and reasoning ; (b) full formalization in first-order logic, including definitions and fact translation ; (c) reasoning conducted purely in natural language; and (d) formal reasoning using a pre-filtered set of facts. This comparison highlights the differences in processing pathways and the structure of the resulting solutions for each approach.

Facts and Hypothesis

Fact1: Benignantness happens.
Fact2: That hands happens results in that that Highlanderness but not enlightening reincarnationism occurs is true.
Fact3: Calculating jailed takes place.
Fact4: If lavishness happens prostitution occurs but fandango does not takes place.
Fact5: That readiness but not arundinaceousness occurs are caused by forgettableness.
Fact6: If trumpeting Asvins takes place then amauroticness happens and outdoing glasnost does not occurs.
Fact7: That fandango takes place yields that lavishness does not takes place.
Fact8: Fandango brings about that prostitution but not lavishness occurs.
Fact9: Combining Yorkshire takes place.
Fact10: Fandango occurs if that closing revisal happens is correct.
Fact11: That lavishness does not occurs holds.
Fact12: Trumpeting Asvins but not stargazinging happens.
Fact13: That quarantining does not occurs leads to that that lavishness occurs and disgruntleding tularaemia does not takes place is right.
Fact14: Quarantining does not happens if that repressing Sclerodermataceae takes place and asterismalness occurs is wrong.

Hypothesis: The statement that prostitution is caused by closing revisal is incorrect."

GPT-4o Response (Non-Necessary case)

To determine if the hypothesis "The statement that prostitution is caused by closing revisal is incorrect" is correct, we must reason using the provided facts.

Step 1: Assume closing revisal happens. (assumption for contradiction)
Step 2: Fact10 states, "Fandango occurs if that closing revisal happens is correct." Thus, if closing revisal happens, then fandango takes place.
...
Step 7: Using int1 and Fact7, it follows that lavishness does not take place. (int3)
Step 8: Fact11 confirms, "That lavishness does not occurs holds," which is consistent with our finding in int3.
Step 9: We proved that fandango happening (due to closing revisal) leads to prostitution happening, supporting step-wise reasoning until prostitution indeed occurs through factual consequences.
Step 10: Therefore, the assumption that prostitution is directly caused by closing revisal through the indirect chain (through fandango) leads to a valid conclusion that prostitution occurs.
Step 11: Hence, there is no contradiction observed that refutes the original claim that closing revisal causes prostitution through the chain found in statements and facts, indirectly.
Based on this reasoning, the hypothesis stating that "prostitution is caused by closing revisal is incorrect" does not hold any demonstrable contradictions and isn't proven false with the current set of facts.

Final conclusion: __DISPROVED__

Figure 5: Case Study: In the GPT-4o response, Step 8 redundantly confirms the conclusion from Step 7 (int3) by citing Fact11, making it an unnecessary step in the reasoning chain. Step 9 is not atomic because it summarizes a multi-step inference (from "fandango happening" to "prostitution happening" using Fact 8) as a single proven outcome without detailing the individual logical operations involved.

Prompt Template: Direct Reasoning

```
Based on the provided facts, answer the question. Conclude with one
of the markers: "__PROVED__" for proven, "__DISPROVED__" for disproven, or
"__UNKNOWN__" if uncertain.
Facts:{facts}
Hypothesis:{hypothesis}
```

Figure 6: Prompt template for direct reasoning. Placeholders: `{facts}`, `{hypothesis}`.

Prompt Template: CoT Reasoning

```
Based on the provided facts, answer the question. Conclude with one
of the markers: "__PROVED__" for proven, "__DISPROVED__" for disproven, or
"__UNKNOWN__" if uncertain.
Facts:{facts}
Hypothesis:{hypothesis}
Let's analyze this step by step.
```

Figure 7: Prompt template for Chain-of-Thought (CoT) reasoning. Placeholders: `{facts}`, `{hypothesis}`.

Prompt Template: Few-Shot Reasoning

```
Based on the provided facts, answer the question. Conclude with one
of the markers: "__PROVED__" for proven, "__DISPROVED__" for disproven, or
"__UNKNOWN__" if uncertain.
Here are some examples of proofs for your reference:
[Start of example]
For example, for this question:
{example}
[End of example]
You can refer to the proof method of the above question, think step by step, and
give the result of this question.
Facts:{facts}
Hypothesis:{hypothesis}
```

Figure 8: Prompt template for few-shot reasoning. Placeholder: `{example}`, `{facts}`, `{hypothesis}`..

Prompt Template: Entity and Predicate Extraction

```
You are a logic analysis expert. Please extract all entities and predicates
from the following logical expression translations:
Translation content: {formula_translations}
facts_formula: {facts_formula}
facts: {facts}
Special Requirement: If any entity or predicate symbol appears in the
facts_formula, but has NO direct definition in the Translation content, you
MUST go to the facts section and locate the corresponding natural language
description and extract it. Be extremely careful NOT to omit any such entities
or predicates. Only skip if it is literally missing from both translation content
and facts.
Task:
1. Identify all entities involved (e.g., this tablefork, this corsair) and
assign variables to them (a, b, c, d...)
2. Identify all predicates (e.g., is a raised, is a collotype) and assign symbols
(using the original symbols like A, B, C...)
Critical instructions:
- Only give full entity and predicate explanations if their definitions appear
in the formula_translations or facts.
- Only include entities and predicates that explicitly appear in the provided
translation content or facts.
- Do not invent, infer, or add any entities or predicates not directly mentioned
in the translations or facts.
- Maintain the original variable identifiers (e.g., 'a' in A(a) corresponds to
the first entity).
- Maintain the original predicate identifiers (e.g., 'A' in A(x) represents "x
is a raised").
- If a symbol (like 'c', 'F', etc.) doesn't appear in the translations or facts,
do not include it in your output.
Expected output format:
We define the entities involved:
- a: [Corresponding entity, e.g., "This tablefork"]
- b: [Corresponding entity, e.g., "This corsair"]...
We denote:
[Original predicate symbol](x): [Predicate description]
[Original predicate symbol](x): [Predicate description]...
Please provide only the requested definitions without any additional information
or explanations.
```

Figure 9: Prompt template for extracting entities and predicates when lowercase variables (entities) are present. Placeholders: {formula_translations}, {facts_formula}, {facts}.

Prompt Template: Predicate Extraction (No Entities)

```
You are a logic analysis expert. Please extract all predicates from the
following logical expression translations:
Translation content: {formula_translations}
facts_formula: {facts_formula}
facts: {facts}
Special Requirement: If any entity or predicate symbol appears in the
facts_formula, but has NO direct definition in the Translation content, you
MUST go to the facts section and locate the corresponding natural language
description and extract it. Be extremely careful NOT to omit any such entities
or predicates. Only skip if it is literally missing from both translation content
and facts.
Task: Identify all predicates and translate each uppercase symbol directly.
Critical instructions:
- For each uppercase symbol in the facts_formula, provide a direct translation
in the format: [SYMBOL]: xxx happened.
- **Do not omit any symbols that appear in facts_formula or translation content.
If they appear, they must be translated.**
- Only include symbols that actually appear in the facts_formula or translation
content.
- Do not invent or infer any entities or relationships not explicitly mentioned.
- If a predicate's meaning is clearly defined in the translations or facts, use
that definition.
- Do not include any lowercase symbols or entity definitions as they are not
relevant in this case.
- If some symbols appear in facts_formula but not in translation content, you
can directly translate the entire formula expression containing those symbols
rather than translating each symbol individually. For example, for an expression
like ¬C→¬(F∧¬E), you don't need to separately translate E if it's not defined
elsewhere.
Expected output format:
We define:
A: xxx happened.
B: xxx happened.
AB: xxx happened...
Please provide only the requested definitions without any additional information
or explanations.
```

Figure 10: Prompt template for extracting predicates when no lowercase variables (entities) are present. Placeholders: {formula_translations}, {facts_formula}, {facts}.

Prompt Template: Logic Proof Translation

```
You are a logic proof translator. Your task is to translate a logical proof sequence from symbolic notation into a clear, step-by-step explanation.
Given: 1. A proof sequence in symbolic form 2. Definitions of entities and predicates used in the proof 3. Logical formula translations
Task: Convert the symbolic proof into a concise, step-by-step explanation that a human can easily follow.
Proof sequence to translate: {proofs_sentence}
Conclusion: {conclusion}
Instructions for translation:
1. Split the proof at each semicolon (;) to identify individual steps.
2. For each step: First, write a brief, natural language explanation on its own line (e.g. "Assume for contradiction: [formula]" or "From [inputs], we derive:"). On the next line, write the step label and the logical formula as in the original proof (e.g. assump1: A(b), int2: ¬B(b), etc.). Do not put both the explanation and the formula on the same line. For assumptions, use "Assume for contradiction: [formula]" then write assumpX: [formula] on the following line. For a standard derived step, use "From [inputs], we derive:" then on the following line write intX: [formula]. For contradictions, use "Contradiction:" then on the following line write "⊥". For reductio ad absurdum, use "By reductio ad absurdum from [step number]:" then write the derived conclusion on the next line. Do not skip formula labels or step names. Write both the explanation and the labeled formula.
3. Maintain correct logical notation (such as ¬, ∧, ∨, →, ∃, ⊥, etc.).
4. In the final step, clearly relate the conclusion to the hypothesis, if appropriate.
5. The output should be only the formatted translation, with no additional commentary.
Output format:
Step 1: [Brief explanation]
[Formula derived]
Step 2: From [input], we derive:
[Formula derived]
Step 3: Assume for contradiction:
assumpX: [Formula derived]
...
{status_message_content}
Final conclusion: {conclusion}
The conclusion must use exactly two underscores before and after either PROVED or DISPROVED or UNKNOWN, with no additional spaces or characters. Translate the proof concisely but retain all logical information from the original proof sequence. Do not add any steps not present in the original, and do not skip any steps. Output the translation only, with no additional commentary.
```

Figure 11: Prompt template for logic proof translation. The placeholder `{proofs_sentence}` is for the symbolic proof sequence. The placeholder `{conclusion}` is for the conclusion (__PROVED__/__DISPROVED__/__UNKNOWN__). The placeholder `{status_message_content}` is replaced by the string 'The search path has been exhausted without finding a way to either prove or disprove the hypothesis.' if `{conclusion}` is '__UNKNOWN__', and is an empty string otherwise (which will result in different spacing around it as per the original prompt generation logic).

Prompt Template: Logical Proof Generation

```
Solve the following logical reasoning problem using formal symbolic logic and
provide a step-by-step reasoning process.
Follow these steps precisely:
1. Define predicates to represent terms in the problem
2. Translate all facts and the hypothesis into formal logical expressions
3. Derive the conclusion through systematic reasoning
4. State the final conclusion
OUTPUT FORMAT:
Your answer should follow this format exactly:
- Begin with "Our problem-solving procedure begins by formalizing all given
facts and the hypothesis into first-order logic using standardized predicate
definitions."
- Then state "For the predicate, we denote:" followed by your predicate
definitions
- Translate each fact into a formal logical expression
- Present your reasoning steps in numbered format (Step 1:, Step 2:, etc.)
- End with "Final conclusion: " followed by either "__PROVED__" or "__DISPROVED__"
IMPORTANT: The conclusion must use exactly two underscores before and after
either PROVED or DISPROVED, with no additional spaces or characters.
Here is an example problem solution, You need to strictly follow the format like
this:
Example Solution:
{fewshot_example}
Now, solve this problem: {question}
The answer should be: {label}
Provide only the solution with no additional commentary or preamble.
```

Figure 12: Prompt template for generating a logical reasoning process. Placeholders: `{question}` for the problem statement, `{label}` for the expected answer (e.g., "__PROVED__"), and `{fewshot_example}` for a formatted example solution.

Prompt Template: Step Validity Evaluation

```
 Premises:
{premises_str}

Conclusion:
{concl_text_full}

Do the premises entail the conclusion? Answer true or false only.
```

Figure 13: Prompt template for evaluating step validity. Placeholders: `{premises_str}` (a string listing the premises, e.g., "fact1: Text of fact 1 int1: Text of intermediate 1"), `{concl_text_full}` (a string representing the conclusion, e.g., "int2: Text of intermediate 2" or "hypothesis: Text of hypothesis"). The model is expected to return 'true' or 'false'.

Prompt Template: Step Atomicity Evaluation

```
 Premises:
{premises_str}

Conclusion:
{concl_text_full}

Is this inference atomic...? Answer true or false only.
```

Figure 14: Prompt template for evaluating step atomicity. Placeholders: `{premises_str}` (a string listing the premises), `{concl_text_full}` (a string representing the conclusion). The model is expected to return 'true' or 'false' indicating if the inference from premises to conclusion is a single, indivisible logical step.