
Empirical Analysis of Chain-of-Thought and Solver-Augmented Large Language Models for Deductive Reasoning

Journal Title
XX(X):2–39
©The Author(s) 2016
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/

SAGE

Ya Wang^{1,2}, Raja Havish Seggoju¹ and Adrian Paschke^{1,2}

Abstract

Large language models (LLMs) have demonstrated remarkable capabilities in multi-step reasoning, yet fundamental questions remain about their ability to perform reliable deductive reasoning. Two contrasting paradigms have emerged: chain-of-thought (CoT) prompting, which treats LLMs as self-contained reasoners, and solver-augmented approaches, which use LLMs as semantic parsers to translate problems into formal logic for symbolic execution. We present a systematic comparison of these approaches under strict single-pass evaluation across three established benchmarks (ProofWriter, PrOntoQA, LogicalDeduction) and controlled synthetic datasets. Our findings reveal three key insights: (1) On standard benchmarks, solver augmentation yields minimal improvements (less than 1%) over zero-shot CoT for state-of-the-art models, [suggesting diminishing performance margins between paradigms under current benchmark settings](#); (2) Mid-sized open-source models (14–32B) match proprietary model performance when solver-augmented, with the primary challenge shifting from generating valid syntax (small models) to accurate semantic parsing (larger models); (3) On synthetic datasets with controlled difficulty variations, solver-augmented methods demonstrate superior robustness, maintaining 76.7% accuracy retention at extreme reasoning depths (14–17 steps) compared to 62.2% for CoT, and showing greater resilience to distracting facts and rules. [These results indicate that while CoT-augmented LLMs achieve strong performance on deductive reasoning tasks up to 5 hops, solver augmentation remains valuable for deeper multi-hop deduction and for applications requiring robust and verifiable reasoning.](#)

Keywords

Hybrid neuro-symbolic systems, Large language models, Chain-of-thought reasoning, Solver augmentation, First-order logic, Deductive reasoning

Introduction

Large language models (LLMs) have transformed natural-language technologies, delivering striking gains in instruction following, tool use, program synthesis, and multi-step problem solving (Chen et al. 2021; Chung et al. 2022; Schick et al. 2023). Much of this progress has come from reasoning-oriented prompting and post-training: chain-of-thought (CoT) prompting and its zero-shot variant elicit stepwise rationales (Wei et al. 2022; Kojima et al. 2022), while supervised and reinforcement-style fine-tuning (SFT/RFT) further amplifies the tendency to generate intermediate steps and verify them internally. Despite these successes, a central question remains: *do LLMs truly perform reliable deductive reasoning?* Early evidence points to important gaps. Creswell et al. (2022) show that models handle single-step inference reasonably well but break down when even basic multi-step chains must be composed. More recently, Liu et al. (2025) report near-random performance on structured deductive reasoning (e.g., syllogisms; Grok3-Think 51.86%, OpenAI o4-mini 54.85%) on multiple-choice questions. Motivated by these findings, we focus this study on first-order logic (FOL) deduction—a setting where conclusions should follow necessarily from explicit premises. Two contrasting paradigms have emerged for enhancing deductive reasoning in LLMs. The first approach, now dominant in industry practice (Fig. 1, right), treats the LLM as a self-contained reasoner: after SFT/RFT on problems with step-by-step solutions, the model generates intermediate tokens and a final answer without external tools. This paradigm assumes that human-like reasoning can emerge from token-to-token generation, rather than relying on explicit search as in classic AI. The second approach, prevalent in recent academic work (Fig. 1, left), treats symbolic solvers as tools and the LLM as a semantic parser that converts natural language problems into formal specifications, typically in first-order logic (FOL)—a representation that LLMs produce more reliably than other logic-programming dialects such as ASP or Pyke (Lam et al. 2024; Beiser et al. 2025). These formalizations are then translated into solver-specific syntax and executed by symbolic engines such as Z3, Prolog, or SAT solvers. Recent systems including Logic-LM (Pan et al. 2023), LINC (Olausson et al. 2023), and CLOVER (Ryu et al. 2024) have demonstrated this approach, reporting substantial gains over CoT prompting on deductive reasoning benchmarks. Having outlined these two paradigms, it is clear that each comes with trade-offs. CoT-style post-training is tool-free, broadly applicable, and easy to deploy; solver-augmented pipelines offer faithfulness,

¹Fraunhofer Institute for Open Communication Systems, DE

²Free University of Berlin, DE

Corresponding author:

Adrian Paschke, Fraunhofer Institute for Open Communication Systems, Kaiserin-Augusta-Allee 31, 10589 Berlin, DE

Email: adrian.paschke@fokus.fraunhofer.de

verifiability, and auditability, but require an external engine and a strict NL-to-logic interface. With rapid improvements in reasoning-centric training, three questions are now open. (i) Do solver-augmented systems still provide advantages on standard FOL deduction benchmarks? (ii) If the LLM’s role is primarily semantic parsing, can smaller open-source models substitute for very large proprietary ones without accuracy loss? (iii) Beyond benchmarks that many models might saturate, how do the two approaches compare under systematically varied structural difficulty, including reasoning depth, distractions, and logical complexity? In this paper, we present a systematic comparison of *CoT-augmented* versus *solver-augmented* LLMs under controlled conditions that ensure fair comparison. We evaluate across three deductive benchmarks (ProofWriter, PrOntoQA, LogicalDeduction) and controlled synthetic datasets that vary reasoning depth and introduce entity-level and rule/fact-level distractions.

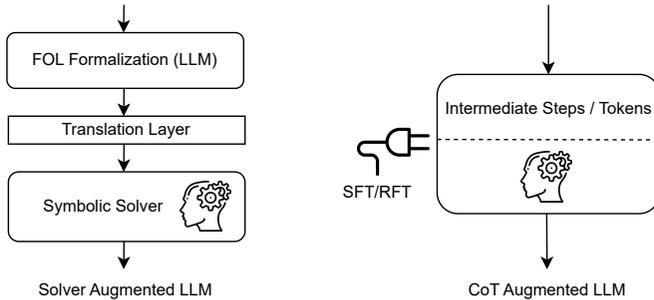


Figure 1. Comparison of solver-augmented and CoT-augmented LLMs. **Left:** In the solver-augmented setting, the LLM first produces a formalization in first-order logic (FOL), which is translated into solver-specific syntax and executed once by a symbolic solver. **Right:** In the CoT-augmented setting, the LLM directly generates intermediate reasoning steps (often enhanced by post-training methods such as supervised or reinforcement fine-tuning), and produces a final answer without invoking an external solver.

Key Findings.

1. **Benchmarks are near ceiling under single-pass evaluation.** On public FOL benchmarks (ProofWriter-Depth 5, PrOntoQA-Depth 5, LogicalDeduction), solver augmentation yields at most small improvements over zero-shot CoT for strong proprietary models (e.g., Gemini 2.5-Pro, Claude-Sonnet-4), typically less than 1 percentage points, and sometimes no gain. [This pattern indicates that, under our single-pass protocol, performance differences between CoT and solver-augmented approaches are marginal on these benchmarks, leaving limited observable headroom for improvement.](#)
2. **Capacity and family effects dominate in the solver pipeline.** Failure modes shift with model capacity: for small models the limiting factor is syntactic acceptance due to invalid NL-to-FOL outputs; for mid-scale models, once the

execution success rate is high, residual errors mainly reflect semantic mismatches in the formalization. Mid-sized open source models (e.g., Qwen3-14/32B, Gemma3-27B) reach near-parity with GPT-5 once they reliably emit valid FOL, whereas smaller DeepSeek-R1 variants frequently fail to produce acceptable formalizations.

3. **Solver augmentation is more robust under controlled difficulty.** On synthetic datasets, solver-augmented models degrade more slowly with increasing reasoning depth (retention 76.7% vs. 62.2% for CoT) and are more resilient to entity substitutions and rule/fact distractors. Accuracy drops with growing structural complexity (numbers of facts, rules, negations) are consistently smaller for the solver pipeline (Figs. 3, 4). Symbolic execution is therefore most valuable in harder cases, such as high-depth chains and noisy contexts, **and is potentially applicable to settings where verifiable reasoning is preferred.**

Related Work

Large language models exhibit stronger multi-step reasoning when encouraged to externalize intermediate steps. Early work (Ling et al. 2017; Nye et al. 2021) demonstrated that generating natural-language rationales can significantly improve program induction and problem solving in mathematical domains. *Chain-of-thought prompting* formalized this idea, showing that giving a few rationale exemplars elicits stepwise deductions across arithmetic, commonsense, and symbolic tasks (Wei et al. 2022); *zero-shot CoT* further showed that a minimal instruction (“let’s think step by step”) can unlock such behavior without exemplars (Kojima et al. 2022). Beyond prompting, various post-training methods have been proposed to further strengthen CoT reasoning. These include supervised and reinforcement learning approaches such as ReFT (Luong et al. 2024), process supervision with intermediate feedback (Lightman et al. 2023), and step-wise reinforcement learning frameworks like STaR (Zelikman et al. 2022).

Recent work also shows that some CoT behavior is already latent in pretrained models and can be revealed by alternative decoding strategies rather than explicit prompting (Wang et al. 2024). A complementary line of work treats symbolic solvers as tools and the LLM as a semantic parser from natural language to formal specifications (e.g., FOL constraints), delegating inference to a solver engine for faithfulness and verifiability. *Logic-LM* (Pan et al. 2023) translates problems to logic and uses solver feedback for iterative self-refinement, reporting sizable gains over CoT on ProofWriter (Tafjord et al. 2021), PrOntoQA (Saparov and He 2022), and related sets. *LINC* (Olausson et al. 2023) similarly parses premises/hypotheses to FOL and offloads deduction to a theorem prover, adding majority-vote mechanisms to improve robustness to parsing errors. *SatLM* (Ye et al. 2023) advocates declarative prompting wherein the LLM specifies constraints while a SAT/SMT/theorem prover computes the answer, mitigating typical planning/calculation errors found in free-form CoT. More recently, *CLOVER* introduced a compositional NL→FOL translation with semantic verification, improving faithfulness on complex sentences and longer reasoning chains (Ryu et al. 2024).

Subsequent work has moved toward more *agentic* LLM–solver interaction: *SymBa* (Lee and Hwang 2025) lets a symbolic backward-chaining solver control proof search via SLD resolution, invoking the LLM only when new premises are needed; *VERUS-LM* (Callewaert et al. 2025) adds semantic satisfiability checks beyond purely syntactic refinement; and *SymbCoT* (Xu et al. 2024) integrates symbolic expressions directly into CoT prompting with a built-in verifier. Wang et al. (2025) further show that adaptively selecting the target formalism (FOL, LP, or SAT) per problem substantially improves solver-augmented accuracy. For a comprehensive survey of these developments, we refer to Cheng et al. (2025) for more detail.

However, both paradigms can be enhanced by auxiliary engineering mechanisms: solver-augmented approaches employ refinement loops, majority voting, or search over candidate formalizations (Pan et al. 2023; Olausson et al. 2023; Ryu et al. 2024), while CoT methods benefit from self-consistency decoding, iterative verification, and ensembling (Wang et al. 2024). While these strategies boost performance, they also add inference-time computation and obscure the intrinsic reasoning contribution of each paradigm. We instead adopt a controlled single-pass protocol: CoT models must output a single label in one forward pass, and solver-augmented models must produce a one-shot formalization executed once by a symbolic engine, without retries or ensembling. By stripping away auxiliary mechanisms from both paradigms, this protocol enables a fair comparison of their intrinsic deductive reasoning capabilities, a controlled evaluation not yet conducted with current state-of-the-art models. To ensure comparability, we restrict tasks to pure first-order logic deduction without fuzzy, modal, or temporal extensions, and we further introduce synthetic variations in reasoning depth and structured distractions to clarify when solver augmentation still provides advantages over modern CoT models.

Experimental Methodology

Study Design and Research Questions

In this study, we systematically compare two contrasting approaches to question answering (QA) on FOL deduction problems: *CoT-augmented LLMs* and *solver-augmented LLMs*. Our goal is to determine whether solver augmentation still provides meaningful benefits over CoT prompting in the era of advanced LLMs, and to identify the conditions under which each approach excels or falters. To achieve this goal, we organize our study around three research questions that address current gaps in the literature:

- **RQ1:** How do CoT-augmented and solver-augmented LLMs perform on FOL-based QA benchmarks when evaluated across state-of-the-art commercial models?
- **RQ2:** Can solver-augmented approaches maintain competitive performance on benchmark datasets when using smaller open-source LLMs (1–32B parameters)?
- **RQ3:** How robust are CoT- and solver-augmented methods to controlled structural variations in synthetic datasets, such as increased reasoning depth, the presence of irrelevant or distracting conditions, and changes in the number of predicates or rules?

These questions are motivated by recent trends in the field: CoT prompting has become the dominant industry practice following extensive post-training on reasoning tasks, raising questions about whether solver-based methods still provide added value; smaller open-source LLMs are increasingly deployed in practice, making solver augmentation potentially attractive for resource-constrained settings; and robustness under controlled conditions remains underexplored in prior work, particularly as reasoning complexity increases beyond benchmark difficulty levels.

Approaches Under Study

CoT-augmented LLMs CoT-augmented LLMs answer questions by directly generating step-by-step solutions, under the assumption that human-like reasoning can emerge from token-to-token generation. In practice, these LLMs are post-trained on problems paired with detailed, step-by-step solutions via supervised fine-tuning (SFT) and/or reinforcement-learning-based fine-tuning (e.g., RLHF), to better elicit such rationales (Nye et al. 2021; Wei et al. 2022; Luong et al. 2024). At inference time, we adopt a *zero-shot CoT* setting: the model receives only a minimal instruction (e.g., “let’s think step by step”) without any in-context examples, and is prompted to generate intermediate natural-language rationales—a *chain of thought (CoT)*—before producing the final answer (see more detailed prompt templates in Appendix, page 24). This setup minimizes prompt requirements and enables us to examine the model’s intrinsic reasoning capabilities under minimal external guidance (Kojima et al. 2022).

Solver-augmented LLMs Solver-augmented LLMs (Pan et al. 2023; Olausson et al. 2023; Ye et al. 2023; Ryu et al. 2024) adopt a hybrid neuro-symbolic design in which the LLM primarily functions as a semantic parser. Given a natural-language question, the model is prompted to abstract the problem into a formal representation consisting of predicates, facts, and rules expressed in first-order logic. To interface with symbolic reasoners, we implement a rule-based translation layer that maps LLM-generated FOL expressions to solver-specific syntax through pattern matching and syntactic transformations. Specifically, this layer handles: (i) conversion of logical connectives (\wedge , \vee , \neg , \rightarrow) to solver-native operators, (ii) transformation of quantifiers (\forall , \exists) into appropriate solver constructs, and (iii) restructuring of implications into Horn clauses for Prolog or assertions for Z3. When the LLM produces syntactically invalid FOL (e.g., malformed predicates or unbalanced parentheses), the system defaults to uniform random selection among answer options, maintaining our strict single-pass evaluation protocol without refinement attempts (see more implementation detail in Appendix, page 26). This design minimizes the LLM’s sensitivity to syntactic variations of solver input languages and isolates its role to semantic abstraction, while delegating logical inference to the symbolic solver. The prompt templates used to guide the semantic parsing process for each dataset are provided in Appendix, page 21.

Datasets

Our study targets QA tasks formulated as FOL deduction, where conclusions follow necessarily from explicitly stated premises. We select datasets with clear, unambiguous

natural-language statements and without modality or fuzzy logic, to isolate deductive reasoning capabilities when comparing CoT- and solver-augmented methods.

Benchmark Datasets Following prior work (Pan et al. 2023; Olausson et al. 2023; Ryu et al. 2024), we evaluate on three established benchmarks where solver-augmented methods have previously shown notable improvements over CoT-based approaches:

- **ProofWriter** (Tafjord et al. 2021): We use the “Depth 5” subset (600 problems), which involves multi-hop implications. Each problem provides a set of premises (facts and rules in natural language) and asks whether a hypothesis is true, false, or undetermined.
- **PrOntoQA** (Saparov and He 2022): We adopt the 5-hop fictional character subset (500 problems), the most challenging split. Each query tests whether a candidate fact follows from a background theory defined in FOL.
- **LogicalDeduction** (Srivastava et al. 2022): We use 300 logic grid puzzles from the BIG-bench “Logical Deduction” task. Each puzzle specifies a set of clues and requires a consistent assignment or ordering.

Synthetic Datasets To probe robustness under controlled structural variations (RQ3), we construct synthetic datasets using the ProverQA pipeline (Qi et al. 2025) with adaptations for pure FOL deduction. Our goal is to systematically vary problem complexity while maintaining logical validity. Public benchmarks provide limited leverage for robustness analysis under our strict single-pass protocol, as strong models already achieve near-ceiling performance and reasoning depth is typically bounded (e.g., ≤ 5 -hop settings). Synthetic instances therefore allow us to isolate individual structural factors and stress-test reasoning behavior beyond the regimes observed in existing benchmarks. We characterize the logical structure of the synthetic data relative to the benchmark datasets in Appendix J.

Generation pipeline. We follow ProverQA’s three-stage procedure: (i) *Background story generation* creates a narrative context; (ii) *Logic skeleton generation* starts from a target goal g and iteratively builds a proof tree, using a symbolic prover to ensure $T \models g$; (iii) *Statement translation* maps the formal representation back to natural language. Each generated instance consists of a logical theory T (facts and rules), a query q , and a gold label $y \in \{\text{True}, \text{False}, \text{Undetermined}\}$, together with metadata describing the reasoning depth and noise configuration. We constrain the logic to pure FOL with standard connectives ($\wedge, \vee, \neg, \rightarrow, \forall, \exists$), restricting rules to definite clauses for decidability. Specifically, rules are expressed in Horn-style form where a conjunction of body atoms implies a single head predicate, ensuring compatibility with symbolic reasoning and efficient satisfiability checking. To improve sample quality, we apply basic sanity filters during generation, including (i) solver executability checks to discard formalizations that cannot be parsed or executed, and (ii) format sanity checks that remove malformed logical structures or incomplete translations. Further implementation details follow the ProverQA framework.

Validation. Label correctness is primarily ensured at the logic-skeleton stage following the ProverQA pipeline: for each generated instance, a symbolic prover assigns and verifies the truth value of the target goal under the assembled premises. We further perform logic validation by executing the prover on the full set of core premises together with injected distractions to confirm that the goal label is preserved. To reduce potential noise introduced during natural-language translation, we additionally apply two consistency checks. First, we compute the answer using the symbolic solver based on the generated formal representation. Second, we obtain the answer predicted by an LLM directly from the natural-language problem statement. Instances for which the solver-derived label and the LLM-predicted label disagree are discarded. Finally, we apply a coverage check to ensure that all entities and predicate terms appearing in the symbolic expressions are also present in the translated natural-language text, following the translation quality-control heuristics described in ProverQA.

Controlled factors. We systematically vary three factors affecting reasoning difficulty while preserving the underlying logical answer.

- **Reasoning depth:** We follow the reasoning-depth categorization used in the ProverQA dataset, where depth is defined as the minimum number of intermediate inference steps required to derive the goal from the premises. The original dataset defines three levels: Easy (1–2 steps), Medium (3–5), and Hard (6–9). To analyze robustness under deeper reasoning regimes, we extend this categorization with two additional levels: Very Hard (10–13) and Extreme (14–17), enabling controlled stress tests beyond the depth range typically present in existing benchmarks.
- **Noise 1 (Entity-level distractions):** Following the distraction mechanism introduced in ProverQA Qi et al. (2025), we randomly duplicate facts/rules with substituted entity names. These distractors share the same predicates but use different entities, making them domain-relevant but task-irrelevant. While ProverQA injects such distractions in a binary manner (present or absent), we extend this design by varying the distraction strength. The parameter $\in \{0.2, 0.4, 0.6, 0.8, 1.0\}$ indicates the proportion of original facts duplicated, allowing us to systematically study how increasing entity-level distractions affect reasoning robustness.
- **Noise 2 (Rule/fact-level distractions):** We add logically valid facts and rules that create alternative reasoning paths unrelated to the main query. These distractors form coherent logical structures but lead to irrelevant conclusions. Similar to Noise 1, ProverQA originally introduces such distractions as a binary condition. We extend this design by controlling the inclusion probability of these distractors, enabling graded difficulty levels in our robustness analysis.

Models

Commercial LLMs. We evaluate four state-of-the-art models: Gemini 2.5-Pro (Google DeepMind 2025), Claude-Sonnet-4 (Anthropic 2025), GPT-5 (OpenAI 2025), and DeepSeek-Reasoner (R1) (DeepSeek 2025). All models are accessed via their respective

APIs with temperature set to 0 to ensure deterministic outputs and reproducibility of results across experimental runs.

Open-source LLMs. For RQ2, we evaluate models from 1B to 32B parameters via `ollama` (Ollama 2025): *Gemma-3* (2B, 9B, 27B), *Qwen-3* (1B, 4B, 7B, 14B, 32B), and *DeepSeek-R1* variants (1.5B, 8B, 14B, 32B). This enables analysis of solver integration across different model capacities.

Symbolic solvers. We employ actively maintained solvers suited to each dataset’s logical structure: *Z3* (de Moura and Bjørner 2008) for PrOntoQA and synthetic datasets (general FOL reasoning), *SWI-Prolog* (Wielemaker et al. 2012) for ProofWriter (Horn clause inference), and the Python *constraint* library (Tauber and contributors 2025) for LogicalDeduction (constraint satisfaction). A rule-based translation layer converts LLM-generated FOL to solver-specific syntax.

Evaluation Protocol

We evaluate performance using multiple metrics to assess both CoT- and solver-augmented approaches. These metrics are chosen to (i) measure end-to-end task performance on the final answer, (ii) decompose solver-augmented failures into *executability vs. formalization correctness*, and (iii) quantify robustness trends under controlled difficulty increases in our synthetic datasets. Following established practice in prior work (Pan et al. 2023; Ryu et al. 2024), we report Overall Accuracy for both settings, and additionally report *Exe_Rate* and *Exe_Acc* for solver-augmented methods to localize error sources. For synthetic controlled factors, we introduce robustness metrics and monotonic trend tests to justify conclusions about degradation patterns rather than single-point scores.

Primary metrics.

- **Overall Accuracy:** Fraction of correct answers across all queries. For solver-augmented methods, failed executions default to uniform random selection among answer options (33.3% expected accuracy for three-choice questions).
- **Execution Success Rate (Exe_Rate):** Fraction of queries with syntactically valid formalizations accepted by the solver (solver-augmented only).
- **Execution Accuracy (Exe_Acc):** Proportion of correct answers among successfully executed cases, serving as an upper bound on answer correctness under solver semantics (solver-augmented only). Note that this metric reflects correctness of the solver-executed formalization and does not guarantee full semantic equivalence between the generated FOL and the intended natural-language meaning.

Robustness metrics. To quantify model robustness under varying conditions (synthetic datasets), we additionally compute:

- **Retention Ratio:** The ratio of accuracy at the most difficult level relative to the easiest level for each experimental dimension (e.g., 14–17 vs. 1–2 reasoning steps, or highest vs. lowest noise), measuring performance degradation.

- **Standard Deviation:** Variance in accuracy across different difficulty levels or noise conditions, indicating stability.
- **Performance Gap (Δ):** The difference between solver-augmented and CoT-augmented accuracy (Solver – CoT), quantifying relative advantage.

These robustness metrics are used only for controlled-factor analyses, where performance should be interpreted as a *curve over difficulty levels*; they capture degradation rate (retention), stability (std), and comparative benefit of solver augmentation (Δ).

Statistical analysis. For synthetic datasets with controlled factors, we compute Spearman correlation coefficients (ρ) to assess monotonic relationships between distraction levels and performance, with two-tailed significance testing (p -values) to validate observed trends. We use Spearman since the controlled factor levels are ordinal (e.g., increasing noise strength), and our goal is to test monotonic degradation rather than assume linearity.

Single-pass inference protocol. We adopt a strict single-pass protocol for both CoT- and solver-augmented settings to isolate the intrinsic deduction capability of the model and the solver, excluding inference-time engineering such as iterative self-refinement, solver-in-the-loop re-prompting, and majority voting. These mechanisms can also benefit CoT-only baselines and substantially confound attribution of gains to the solver itself. Table 2 (Appendix) summarizes the inference-time mechanisms used by prior solver-augmented systems, which typically combine few-shot prompting with refinement and/or voting and therefore report larger solver–CoT gaps than a single-pass evaluation.

Experimental setup. For fair comparison, we enforce standardized formats: both approaches output from {A: True, B: False, C: Undetermined}. CoT models receive zero-shot prompts with explicit format instructions. Solver-augmented models receive prompts with logical syntax specifications and three illustrative examples to guide formalization. All experiments use greedy decoding with single-pass evaluation to reflect practical deployment scenarios. We report results averaged across all test instances for each dataset, with additional stratified analysis by structural complexity factors where applicable.

Results & Discussion

RQ1: CoT-Augmented vs Solver-Augmented Performance

Our results reveal that solver-augmented approaches no longer deliver the substantial gains on these three benchmarks that were reported in earlier work (Pan et al. 2023; Olausson et al. 2023; Ryu et al. 2024). For Gemini-2.5-Pro, Claude-Sonnet-4, DeepSeek-Reasoner and GPT-5, solver-augmented pipelines improve accuracy by less than 1% on average (Table 1). These modest gains are primarily observed on ProofWriter, where solver augmentation yields small improvements across all models (+0.06% to +2.23%). PrOntoQA shows no differentiation, as both methods reach ceiling performance. On LogicalDeduction, solver-augmented pipelines underperform CoT across all four models

(−0.33% to −1.00%), resulting in near-zero or slightly negative average Δ for Claude-Sonnet-4 (−0.25%) and GPT-5 (−0.13%). Notably, several of these differences fall within variance—for instance, GPT-5’s $\Delta = +0.06$ pp on ProofWriter lies well within the CoT standard deviation of ± 0.22 pp (Table 4)—suggesting they are not statistically distinguishable from noise. Further analysis of error patterns reveals distinct failure modes: solver-augmented errors primarily stem from duplicate predicate definitions during formalization, while CoT errors arise from occasional mistakes at individual reasoning steps (e.g., misinterpreting a premise or incorrectly applying a rule) rather than a complete inability to perform logical reasoning (see more detailed case studies in Appendix I).

Table 1. Accuracy comparison of CoT- and Solver-augmented approaches across three reasoning datasets. All values are the mean accuracy over three runs. Δ denotes the accuracy difference (Solver – CoT) in percentage points. Avg. Δ denotes the mean improvement across the three datasets. PrOntoQA reaches ceiling performance (100%) for both methods, hence $\Delta = 0$ and contributes neutrally to the average. Positive values indicate that Solver outperforms CoT. Full results across three runs are reported in Appendix Table 4. A comparison between zero-shot and few-shot CoT prompting is provided in Table 5, while additional model comparisons are presented in Table 3.

Model	ProofWriter			PrOntoQA			LogicalDeduction			Avg. Δ (%)
	CoT(%)	Solver(%)	Δ (%)	CoT(%)	Solver(%)	Δ (%)	CoT(%)	Solver(%)	Δ (%)	
Gemini-2.5-pro	96.43	98.66	+2.23	100.00	100.00	0.00	100.00	99.67	−0.33	+0.63
Claude-Sonnet-4	98.40	98.66	+0.26	100.00	100.00	0.00	99.67	98.67	−1.00	−0.25
Deepseek-reasoner	96.87	98.61	+1.74	100.00	100.00	0.00	100.00	99.33	−0.67	+0.36
GPT-5	98.60	98.66	+0.06	100.00	100.00	0.00	100.00	99.56	−0.44	−0.13

This stands in sharp contrast to prior studies, where solver augmentation produced substantial improvements: Logic-LM (Pan et al. 2023) using GPT-4 reported gains of +11.55% and +12.38% on ProofWriter and LogicalDeduction, while CLOVER (Ryu et al. 2024) using GPT-4o observed improvements of +18.7% and +5.3%. Our results indicate that recent post-training practices, such as chain-of-thought fine-tuning, have substantially enhanced models’ deductive reasoning capabilities, effectively narrowing the gap between pure CoT and solver-augmented methods on public benchmarks.

These findings raise two important questions that we address in subsequent experiments. First, if solver-augmented pipelines primarily require the LLM to perform semantic parsing, then smaller open-source models might suffice—potentially replacing very large proprietary models without significant accuracy loss (explored in RQ2). Second, the near-perfect performance of strong LLMs on current benchmarks suggests limited headroom for distinguishing reasoning approaches: these datasets may be too simple or too familiar to the models, potentially masking differences in robustness. To mitigate this limitation, we complement the benchmark evaluation with controlled synthetic datasets (RQ3) that systematically increase reasoning depth beyond the 5-hop ceiling of ProofWriter and PrOntoQA and introduce targeted perturbations (Noise 1/2 and structural complexity). These experiments allow us to examine model behavior in settings where benchmark ceiling effects no longer dominate.

RQ2: Impact of Model Scale and Family

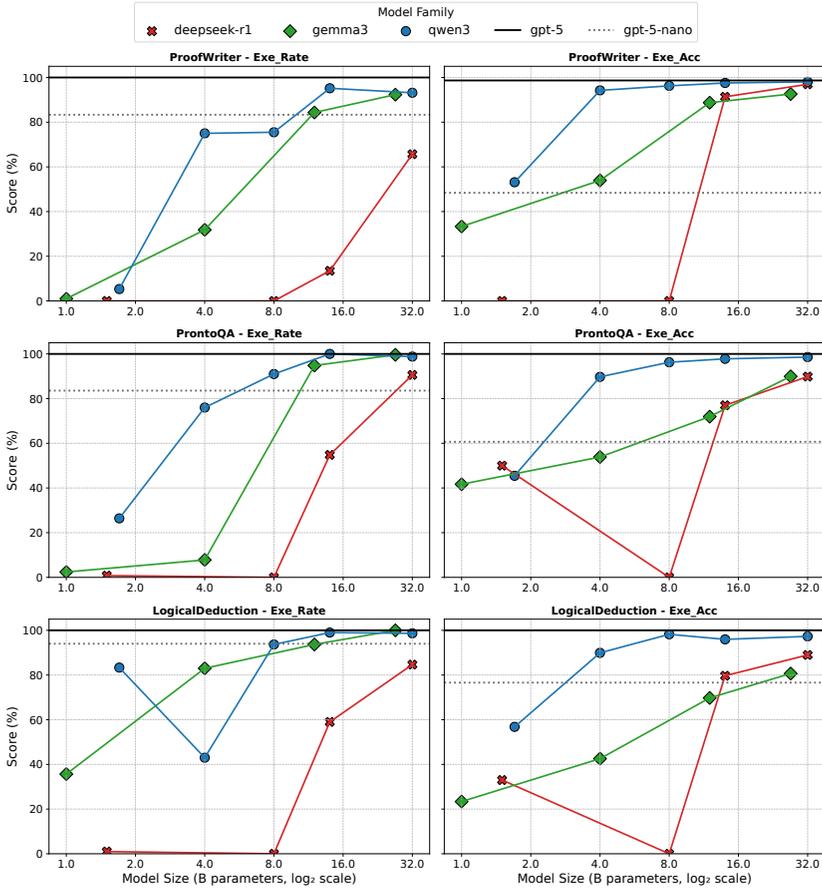


Figure 2. Solver-augmented performance vs. model scale on *ProofWriter*, *PrOntoQA* and *LogicalDeduction*. **Left:** Execution success rate (syntactic acceptance by the solver). **Right:** Execution accuracy (correct final answer after solver execution). Higher values indicate better performance. Model families: Gemma3 (green \blacklozenge), Qwen3 (blue \bullet), DeepSeek-R1 (red \times); GPT-5 and GPT-5-nano shown as horizontal reference baselines (gray dashed lines). See additional experiments with the CoT-augmented settings in Appendix Fig. 5.

We next investigate how model scale and family affect performance in the solver-augmented setting, focusing on open-source LLMs ranging from 1B to 32B parameters. Figure 2 summarizes results on *ProofWriter*, *PrOntoQA*, and *LogicalDeduction*, reporting both execution success rate (syntactic acceptance by the solver) and execution accuracy (correct final answer after solver execution). GPT-5 and GPT-5-nano serve as horizontal reference baselines.

Scaling trends. We evaluate released open-source models without additional fine-tuning to isolate their intrinsic ability to follow formalization prompts under solver augmentation. Across all three datasets, we observe consistent scaling behavior: larger models achieve higher execution rates and accuracy, with performance steadily approaching the GPT-5 baseline. Small models (1–4B) frequently fail to generate syntactically valid formalizations, resulting in very low execution rates that cap their execution accuracy. In contrast, mid- to large-scale models (14–32B) achieve near-ceiling accuracy (exceeding 95%), demonstrating that solver augmentation enables them to match state-of-the-art proprietary models.

Model family differences. Among the evaluated families, *Qwen3* (blue circles) demonstrates the strongest performance: even at 4B parameters it surpasses 90% execution accuracy on ProofWriter and PrOntoQA, and at 32B it essentially matches GPT-5. *Gemma3* (green diamonds) shows more gradual improvement, requiring 12–27B parameters to achieve competitive performance, though it ultimately reaches similar accuracy levels. *DeepSeek-R1* (red crosses) exhibits the weakest scaling trajectory: while the 14B and 32B versions eventually achieve approximately 90% execution accuracy, smaller variants (1.5B and 8B) completely fail with execution rates approaching zero, rendering them unsuitable for this pipeline.

Baseline comparison and failure analysis. Our results demonstrate that mid-sized open-source models (*Qwen3*-14B/32B, *Gemma3*-27B) achieve near-parity with GPT-5 on deductive reasoning benchmarks in the solver-augmented setting, indicating that very large proprietary models are not strictly necessary for strong performance. However, we observe a complete breakdown in smaller *DeepSeek-R1* models: despite explicit instructions to output FOL formalizations, they persistently attempt to solve problems directly in natural language, producing invalid solver inputs. We hypothesize that this behavior stems from *DeepSeek-R1*’s training methodology. According to the technical report (Guo and et al. 2025), the initial *DeepSeek-R1-Zero* was optimized primarily through reinforcement learning without strong instruction-following supervision, resulting in poor readability and structural non-compliance. Although later stages incorporated cold-start supervised fine-tuning and format rewards, these signals appear insufficient for small-scale variants, leaving their instruction-following capabilities fragile.

These findings support the conclusion that mid-sized open source models can effectively substitute for very large ones in solver-augmented settings, while also highlighting that reliable solver integration depends critically on robust instruction-following alignment. Moreover, fine-tuning such models specifically on natural language to FOL mapping tasks could further enhance their semantic parsing capabilities.

RQ3: Performance on Synthetic Datasets

Experiments with Controlled Factors

Impact of reasoning depth. Figure 3 (left) reports accuracy as a function of reasoning steps, ranging from 1–2 to 14–17. Both CoT and solver-augmented models exhibit

clear monotonic decline as depth increases, though the solver pipeline degrades more gradually. In the shallow regime (1–5 steps), the two approaches perform nearly identically, maintaining accuracy above 90%. Notably, benchmark datasets such as ProofWriter and PrOntoQA are limited to 5-hop reasoning, confirming that within this range CoT-augmented methods can match solver-augmented performance. Beyond 5 steps, however, the gap widens substantially: at 6–9 steps the solver maintains a +5% advantage, increasing to +8% at 10–13 steps and +14% at 14–17 steps. To better understand the failure patterns of CoT reasoning at greater depths, we conducted an additional error analysis of incorrect CoT outputs. A systematic annotation of failure types is reported in Appendix K. The analysis indicates that deeper CoT failures are primarily associated with reasoning errors (i.e., incorrect logical inferences or misinterpretations of the given premises) rather than the introduction of hallucinated facts or premises not supported by the input. Qualitative inspection (detailed in Appendix I) further suggests that CoT outputs can become internally inconsistent in deeper reasoning chains. Models may produce self-contradictory intermediate steps or misapply assumptions. Solver-augmented models also show accuracy degradation with depth, but for different reasons. As reasoning steps increase, the semantic parser faces growing difficulty in producing correct formalizations—for instance, in defining and consistently reusing predicates or in correctly instantiating rules with appropriate arguments. These parsing errors accumulate with increasing reasoning depth and reduce execution accuracy, even though the downstream solver itself remains logically sound. To quantify robustness, we compute a retention ratio: the accuracy at the deepest level (14–17 steps) relative to the shallowest level (1–2 steps). The solver retains 76.7% of its initial performance, compared to only 62.2% for CoT. Additionally, the solver demonstrates lower variance across depth bins ($\sigma = 8.7$ vs. 13.6), indicating more stable degradation. These results suggest that while CoT post-training enables strong shallow reasoning up to 5 hops, solver augmentation provides superior stability as reasoning depth increases.

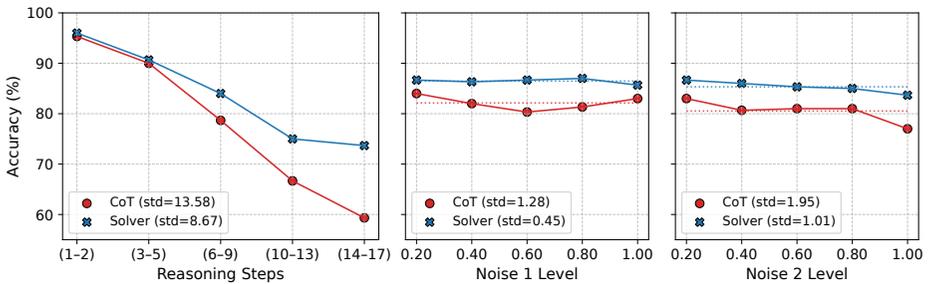


Figure 3. Comparison of CoT and Solver accuracies across different conditions. The left panel shows performance with varying reasoning step difficulty, while the middle and right panels present results under increasing noise levels (Noise 1 and Noise 2). Dashed lines indicate mean values. The corresponding numerical results are provided in Appendix Tables 6, 7, and 8.

Impact of entity-level distractions (Noise 1). To further probe robustness, we apply name substitution as entity-level noise on the reasoning depth 6–9 subset, generating five variants with substitution rates from 0.2 to 1.0. As shown in Figure 3 (middle), both pipelines remain relatively stable across noise levels. CoT accuracy fluctuates between 80.3–84.0% ($\mu = 82.1\%$, $\sigma = 1.3\%$), while the solver achieves consistently higher performance (85.7–87.0%, $\mu = 86.5\%$, $\sigma = 0.5\%$), maintaining an average margin of +4.3%. Importantly, no monotonic degradation occurs as noise increases: Spearman correlations are weak and statistically insignificant (CoT: $\rho = -0.30$, $p = 0.62$; Solver: $\rho = -0.21$, $p = 0.74$), with retention ratios at noise level 1.0 relative to 0.2 remaining nearly identical for both methods (≈ 0.99). The minimal impact of entity-level distractions indicates that both CoT and solver-augmented methods can effectively filter out irrelevant name variations, focusing on the logical relationships rather than being confused by the presence of similar facts about different entities.

Impact of rule/fact-level distractions (Noise 2). We next examine robustness under distracting rules and facts, again using the reasoning depth 6–9 subset. Figure 3 (right) reveals that performance decreases as the probability of injecting distractors increases. CoT accuracy drops from 83.0% at noise level 0.2 to 77.0% at 1.0 ($\mu = 80.5\%$, $\sigma = 1.95\%$, retention = 0.93), indicating that distracting rules and facts systematically interfere with natural-language reasoning chains. Solver-augmented models decline more gently, from 86.7% to 83.7% ($\mu = 85.3\%$, $\sigma = 1.01\%$, retention = 0.97), with strictly monotonic decrease (Spearman $\rho = -1.00$, $p = 0.0167$). The smaller degradation suggests that while additional distractors complicate semantic parsing into FOL, once a consistent formalization is achieved, the solver can effectively filter irrelevant information and preserve correctness. Across all noise levels, the solver consistently outperforms CoT by 3.7–6.7%, demonstrating superior robustness to rule/fact-level distractions.

Effect of Structural Complexity To analyze the impact of structural complexity beyond controlled synthetic settings, we aggregate all instances from the synthetic datasets and stratify them by the number of facts, rules, and negations. For each measure, we partition the combined dataset into five quantiles, yielding balanced subsets that enable examination of how increasing structural complexity correlates with model accuracy (see Appendix Tables 9, 10, and 11 for details).

Figure 4 summarizes the effects of varying structural elements. For *facts*, accuracy decreases steadily as premises increase: CoT drops from 88.8% to 68.7% ($\sigma = 6.6\%$), while the solver declines more gradually from 90.7% to 78.1% ($\sigma = 4.1\%$). The performance gap widens with scale, indicating that CoT struggles to maintain coherent chains in longer contexts, whereas the solver—despite increased formalization burden—benefits from symbolic inference. For *negations*, CoT shows moderate but non-monotonic decline from 84.9% to 75.6% (approximately 1.4% per quantile step). The solver remains comparatively stable, decreasing from 87.0% to 82.7% (approximately 0.2% per step). The performance gap between solver and CoT widens as negations increase (from +2.1% to +7.4%), with higher retention for the solver (0.95 vs. 0.89). The strongest effect emerges with *rules*: CoT accuracy falls sharply from 89.0% to 65.1%

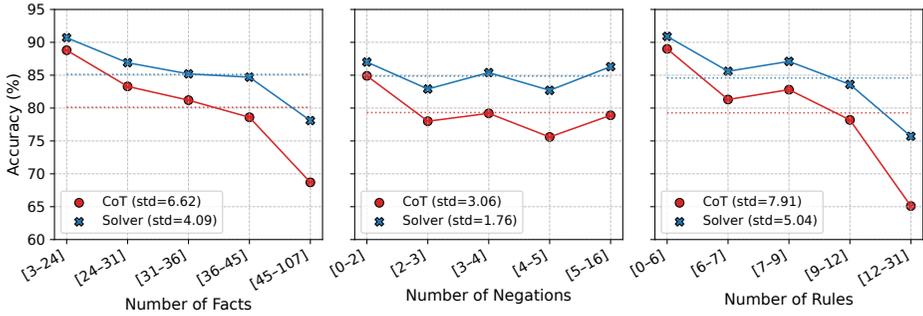


Figure 4. Accuracy comparison between CoT and Solver across different ranges of (left) number of facts, (middle) number of negations, and (right) number of rules. Dashed lines indicate mean values across all quantiles.

($\sigma = 7.9\%$), while the solver drops from 90.9% to 75.7% ($\sigma = 5.0\%$). This indicates that increasing numbers of rules are particularly disruptive for CoT’s language-based reasoning, whereas solver-augmented pipelines maintain higher accuracy and lower variance. We note that this analysis is observational, as these structural factors are not fully independent, and thus reflects correlations rather than causal effects.

Limitations

Our study has three primary limitations. First, we restrict evaluation to pure first-order logic, excluding modal, temporal, and probabilistic reasoning common in real-world applications. The near-parity between CoT and solver-augmented methods we observe on FOL benchmarks may not hold for these richer logical frameworks. Second, our datasets—both benchmarks and synthetic—use clear, unambiguous problem statements with simple sentence structures. Real-world deductive reasoning involves complex natural language with nested clauses, ambiguous references, and context-dependent interpretations that pose different challenges for each approach. While our synthetic datasets test robustness to entity substitutions and distracting facts, they maintain straightforward linguistic presentation. Natural language complexities such as pronominal resolution, scope ambiguity, and syntactic garden paths could create asymmetric difficulties for the two approaches, though which method would be more affected remains an open question without empirical evaluation. **Third, we adopt a strict single-pass inference protocol to isolate intrinsic reasoning and solver contributions.** While this controlled setup improves internal validity and attribution clarity, it may underestimate the absolute performance achievable under multi-pass refinement, solver-in-the-loop re-prompting, or majority voting pipelines commonly used in prior work. Such mechanisms can increase solver-augmented gains, but they introduce additional computation and confound attribution of improvements to the solver itself. Our reported results should therefore be interpreted as conservative estimates of solver impact under minimal inference-time engineering rather than upper-bound system performance.

Conclusion & Future Work

Conclusion

This work presents a systematic comparison of CoT-augmented versus solver-augmented LLMs for first-order logic deduction under strict single-pass evaluation. Our findings reveal that the performance gap between these approaches has **substantially** narrowed on standard benchmarks, with state-of-the-art models achieving near-ceiling accuracy (**fewer than 1 percentage point** of improvement from solver augmentation). This convergence suggests that recent advances in reasoning-oriented post-training have substantially enhanced LLMs' deductive capabilities for shallow reasoning tasks. However, this apparent parity masks important differences under challenging conditions. While both approaches perform comparably on problems requiring up to 5 reasoning steps, solver-augmented methods demonstrate superior robustness at extreme depths, retaining 76.7% accuracy at 14–17 steps compared to 62.2% for CoT. This differential degradation reveals that pure neural approaches suffer from compounding reasoning errors in long reasoning chains, whereas solver augmentation maintains relative stability by delegating inference to sound symbolic engines. Our scaling analysis reveals that mid-sized open-source models (14–32B parameters) achieve near-parity with GPT-5 in solver-augmented settings, as their task reduces to semantic parsing rather than end-to-end reasoning. This accessibility enables resource-constrained deployments without sacrificing accuracy on complex deductive tasks. **Taken together, these** results indicate that while CoT post-training has achieved remarkable progress, solver augmentation remains valuable for (i) complex multi-hop inference beyond 5 steps, (ii) resource-efficient deployments with smaller models, and (iii) **scenarios requiring robustness under increasing reasoning complexity and formally verifiable inference**.

Future Work

Several directions emerge from this study. First, extending beyond first-order logic to modal, temporal, and probabilistic reasoning would reveal whether **the benefits of** symbolic augmentation generalize across reasoning types. Second, our finding that mid-sized models effectively serve as semantic parsers suggests opportunities for specialized fine-tuning on NL-to-FOL translation, potentially yielding efficient “reasoning modules” that combine smaller model accessibility with formal method reliability. Third, the complementary failure modes—hallucination for CoT versus parsing errors for solvers—motivate hybrid architectures that could leverage both strengths. Potential designs include using CoT to guide formal translation or adaptive systems that dynamically select approaches based on problem characteristics. Fourth, beyond execution accuracy, future work could develop more fine-grained metrics for formalization quality, **such as** semantic equivalence and faithfulness between generated FOL and the intended natural-language meaning. Such evaluation may combine limited human annotation, formal equivalence checks **where** applicable, and LLM-assisted assessment as complementary signals, while ensuring **the** reliability of the **evaluation framework** itself. Finally, our results highlight the need for more challenging benchmarks **featuring** deeper reasoning chains, complex

quantifier structures (nested and alternating quantifiers), and realistic noise. As LLMs continue to improve, solver-based methods may shift from [general-purpose](#) augmentation to specialized tools for particular problem classes, [yet](#) their contribution to robust, trustworthy AI systems remains essential.

References

- Anthropic (2025) Introducing claude 4 (opus 4 and sonnet 4). <https://www.anthropic.com/news/claude-4>. Announcement of Claude Sonnet 4 and Opus 4.
- Beiser A, Penz D and Musliu N (2025) Intermediate languages matter: Formal languages and llms affect neurosymbolic reasoning. *arXiv preprint arXiv:2509.04083* .
- Callewaert B, Vandeveld S and Vennkens J (2025) VERUS-LM: A versatile framework for combining LLMs with symbolic reasoning. *arXiv preprint arXiv:2501.14540* URL <https://arxiv.org/abs/2501.14540>.
- Chen M, Tworek J, Jun H, Yuan Q, Ponde H and et al (2021) Evaluating large language models trained on code. In: *NeurIPS*.
- Cheng F, Li H, Liu F, Van Rooij R, Zhang K and Lin Z (2025) Empowering llms with logical reasoning: A comprehensive survey. *arXiv preprint arXiv:2502.15652* .
- Chung HW, Hou L, Longpre S, Zoph B, Tay Y, Fedus W, Li E, Wang X, Dehghani M, Brahma S, Webson A et al. (2022) Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416* .
- Creswell A, Shanahan M and Higgins I (2022) Selection-inference: Exploiting large language models for interpretable logical reasoning. *arXiv preprint arXiv:2205.09712* .
- de Moura L and Bjørner N (2008) Z3: An efficient smt solver. In: *Tools and Algorithms for the Construction and Analysis of Systems (TACAS), Lecture Notes in Computer Science*, volume 4963. Springer, pp. 337–340. DOI:10.1007/978-3-540-78800-3_24. URL https://doi.org/10.1007/978-3-540-78800-3_24.
- DeepSeek (2025) Deepseek-r1 release. <https://api-docs.deepseek.com/news/news250120>. Initial release post; open-source technical report and models.
- Google DeepMind (2025) Gemini 2.5 pro: Our most intelligent ai model. <https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/>. Announces availability of Gemini 2.5 Pro in AI Studio/Gemini app.
- Guo D and et al (2025) Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948* URL <https://arxiv.org/abs/2501.12948>.
- Kojima T, Gu SS, Reid M, Matsuo Y and Iwasawa Y (2022) Large language models are zero-shot reasoners. *Advances in neural information processing systems* 35: 22199–22213.
- Lam LHM, Thatikonda RK and Shareghi E (2024) A closer look at tool-based logical reasoning with llms: The choice of tool matters. In: *Proceedings of the 22nd Annual Workshop of the Australasian Language Technology Association*. pp. 41–63.
- Lee J and Hwang W (2025) SymBa: Symbolic backward chaining for structured natural language reasoning. In: *Proceedings of the 2025 Conference of the Nations of the Americas Chapter*

of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers). Association for Computational Linguistics, pp. 2468–2484. DOI:10.18653/v1/2025.naacl-long.124. URL <https://aclanthology.org/2025.naacl-long.124/>.

- Lightman S, Heimersheim S, Wu T, Askell A, Henighan T, Chen A, Glaese A, Irving G, McAleese NJ, Kravec S et al. (2023) Let’s verify step by step. *arXiv preprint arXiv:2305.20050* URL <https://arxiv.org/abs/2305.20050>.
- Ling W, Yogatama D, Dyer C and Blunsom P (2017) Program induction by rationale generation: Learning to solve and explain algebraic word problems. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL), Long Papers*. Vancouver, Canada: Association for Computational Linguistics, pp. 158–167. DOI:10.18653/v1/P17-1015. URL <https://aclanthology.org/P17-1015/>.
- Liu H, Ding Y, Fu Z, Zhang C, Liu X and Zhang Y (2025) Evaluating the logical reasoning abilities of large reasoning models. *arXiv preprint arXiv:2505.11854*.
- Luong TQ, Zhang X, Jie Z, Sun PK, Jin X and Li H (2024) Reft: Reasoning with reinforced fine-tuning. In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*. URL <https://aclanthology.org/2024.acl-long.410/>.
- Nye M, Andreassen AJ, Gur-Ari G, Michalewski H, Austin J, Bieber D, Dohan D, Lewkowycz A, Bosma M, Luan D, Sutton C and Odena A (2021) Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114* URL <https://arxiv.org/abs/2112.00114>.
- Olausson TX, Gu A, Lipkin B, Zhang CE, Solar-Lezama A, Tenenbaum JB and Levy R (2023) Linc: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. URL <https://aclanthology.org/2023.emnlp-main.313/>.
- Ollama (2025) Ollama: Run large language models locally. <https://ollama.com/>. Official website and docs hub.
- OpenAI (2025) Introducing GPT-5. <https://openai.com/index/introducing-gpt-5/>. Unified system with deeper reasoning mode (GPT-5 thinking).
- Pan L, Albalak A, Wang X and Wang WY (2023) Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. *arXiv preprint arXiv:2305.12295*.
- Qi C, Ma R, Li B, Du H, Hui B, Wu J, Laili Y and He C (2025) Large language models meet symbolic provers for logical reasoning evaluation. *arXiv preprint arXiv:2502.06563*.
- Ryu H et al. (2024) Divide and translate: Compositional first-order logic translation and verification for complex logical reasoning (clover). *arXiv preprint arXiv:2410.08047* URL <https://arxiv.org/abs/2410.08047>. ICLR 2025 (authors’ repository).
- Saparov A and He H (2022) Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *arXiv preprint arXiv:2210.01240*.
- Schick T, Dwivedi-Yu J, Dessi R, Raileanu R, Lomeli M, Bach SH, Goyal N, Huang XL, Mazaré PE, Suhr A et al. (2023) Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

- Srivastava A, Rastogi A, Rao A and et al (2022) Beyond the imitation game benchmark (big-bench). *arXiv preprint arXiv:2206.04615* .
- Tafjord Ø, Mishra BD and Clark P (2021) Proofwriter: Generating implications, proofs, and abductive statements over natural language. In: *Findings of the Association for Computational Linguistics (ACL)*.
- Tauber J and contributors (2025) python-constraint: Constraint solving problem module for python. <https://python-constraint.github.io/python-constraint/>. Accessed: 2025-09-26.
- Wang X, Yang H, Cheng F and Liu F (2025) Adaptive selection of symbolic languages for improving LLM logical reasoning. *arXiv preprint arXiv:2510.10703* URL <https://arxiv.org/abs/2510.10703>.
- Wang X et al. (2024) Chain-of-thought reasoning without prompting. *arXiv preprint arXiv:2402.10200* URL <https://arxiv.org/abs/2402.10200>. NeurIPS 2024 poster.
- Wei J, Wang X, Schuurmans D, Bosma M, Ichter B, Xia F, Chi EH, Le QV and Zhou D (2022) Chain-of-thought prompting elicits reasoning in large language models. In: *Advances in Neural Information Processing Systems (NeurIPS)*. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.
- Wielemaker J, Schrijvers T, Triska M and Lager T (2012) Swi-prolog: description and overview. *Theory and Practice of Logic Programming* 12(1–2): 67–96. DOI:10.1017/S1471068411000494. URL <https://doi.org/10.1017/S1471068411000494>.
- Xu J, Fei H, Pan L, Liu Q, Lee ML and Hsu W (2024) Faithful logical reasoning via symbolic chain-of-thought. In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Bangkok, Thailand: Association for Computational Linguistics, pp. 13326–13365. DOI:10.18653/v1/2024.acl-long.720. URL <https://aclanthology.org/2024.acl-long.720/>.
- Ye X, Chen Q, Dillig I and Durrett G (2023) Satlm: Satisfiability-aided language models using declarative prompting. In: *Advances in Neural Information Processing Systems (NeurIPS)*. URL <https://arxiv.org/abs/2305.09656>.
- Zelikman E, Wu Y, Mu J and Goodman ND (2022) Star: Bootstrapping reasoning with reasoning. *arXiv preprint arXiv:2203.14465* URL <https://arxiv.org/abs/2203.14465>.

Appendix

A.1: Prompt Templates for Solver-Augmented Approach

PrOntoQA PrOntoQA uses standard first-order logic notation with universal quantifiers for fictional ontologies:

Task:

You are a logic translator. Your task is to convert the given Problem and Question into First-Order Logic (FOL).

Important Instructions:

- Follow the formatting **exactly**: use triple colons (:::)
- The Formulas must only use Predicates defined in the Predicates section
- Do not include reasoning or extra text

Example:

Problem: Each jompus is fruity. Every jompus is a wumpus. Every wumpus is not transparent. [...] Alex is a tumpus.

Question: True or false: Alex is not shy.

Output Format:

Predicates:

jompus(x) ::: x is a jompus

fruity(x) ::: x is fruity

wumpus(x) ::: x is a wumpus

[...additional predicates...]

Formulas:

$\forall x (jompus(x) \rightarrow fruity(x))$::: Each jompus is fruity.

$\forall x (jompus(x) \rightarrow wumpus(x))$::: Every jompus is a wumpus.

$\forall x (wumpus(x) \rightarrow \neg transparent(x))$::: Every wumpus is not transparent.

tumpus(alex) ::: Alex is a tumpus.

[...additional formulas...]

Query:

$\neg shy(alex)$::: Alex is not shy

ProverQA (Synthetic Dataset) ProverQA requires careful handling of universal versus specific rules:

Task:

You are a logic translator. First define all Predicates clearly, then convert Problem and Question into First-Order Logic.

Universal vs Specific Rules:

- "If a/an [Subject_category]...", "All [Subject_category] who..."
→ Universal rules with $\forall x$ (do NOT create predicate for Subject_category)
- "If [Name]...", "When [Name]..."
→ Specific rules using the name in every atom, no quantifier

Example with Subject_category: writer

Problem: If a writer expresses well, then they share ideas. Chloe does not communicate clearly. [...]

Question: Based on the above information, is the following statement true, false, or uncertain? Lee inspires readers.

Output:

Predicates:

expresses_well(x) ::: x expresses well
 shares_ideas(x) ::: x shares ideas
 [...additional predicates...]

Formulas:

$\forall x$ (expresses_well(x) \rightarrow shares_ideas(x)) ::: If a writer expresses well, then they share ideas.
 \neg communicates_clearly(Chloe) ::: Chloe does not communicate clearly.
 writes_cleverly(Lee) \vee master_brevity(Lee) ::: Lee writes cleverly or masters brevity.
 [...additional formulas...]

Query:

inspire_readers(Lee) ::: Lee inspires readers

ProofWriter For ProofWriter, we employ a Prolog-style representation with custom notation for negation to maintain compatibility with our solver pipeline:

Task Description:

You are given a problem description and a question. The task is to:

- 1) define all the predicates in the problem
- 2) parse the problem into logic rules based on the defined predicates
- 3) write all the facts mentioned in the problem
- 4) parse the question into the logic form

Important Instructions:

- Follow the formatting **exactly** as in the examples
- **Predicate names & constants are lowercase**; variables start with "\$"
- To express negation, use the predicate with second argument set to False (e.g., Kind(x, False) means "x is not kind")
- **Do NOT use prefix-style negation such as 'not', '~', or '!'**

Example:

Problem: Anne is quiet. Erin is furry. [. . .] All red people are young.

Question: Based on the above information, is the following statement true, false, or unknown? Anne is white.

Output Format:

Predicates:

```
quiet($x, bool) ::: is $x quiet?
furry($x, bool) ::: is $x furry?
[...additional predicates...]
```

Facts:

```
quiet(anne, True) ::: anne is quiet.
furry(erin, True) ::: erin is furry.
[...additional facts...]
```

Rules:

```
young($x, True) >> furry($x, True) ::: young people are furry.
red($x, True) >> young($x, True) ::: all red people are young.
[...additional rules...]
```

Query:

```
white(anne, True) ::: anne is white.
```

LogicalDeduction For constraint satisfaction problems in LogicalDeduction, we define domains, variables, and constraints:

Task Description:

Parse the problem as a constraint satisfaction problem, defining the domain, variables, and constraints.

Format Specification:

Domain:

1: leftmost/oldest

N: rightmost/newest

Variables:

entity_name [IN] [1, 2, ..., N]

Constraints:

entity == value ::: Natural language justification

entity1 < entity2 ::: Natural language justification

AllDifferentConstraint([...]) ::: All entities have different values

Example:

Problem: In an antique car show, there are three vehicles: a station wagon, a convertible, and a minivan. The station wagon is the oldest. The minivan is newer than the convertible.

Output:

Domain:

1: oldest

3: newest

Variables:

station_wagon [IN] [1, 2, 3]

convertible [IN] [1, 2, 3]

minivan [IN] [1, 2, 3]

Constraints:

station_wagon == 1 ::: The station wagon is the oldest.

minivan > convertible ::: The minivan is newer than the convertible.

AllDifferentConstraint([station_wagon, convertible, minivan]) ::: All vehicles have different values.

A.2: Prompt Template for CoT-Augmented Model

We make use of a uniform template across datasets that uses the same zero-shot approach.

Solve this problem step by step. Think through your reasoning carefully, then provide your final answer as a single letter ([[OPTIONS_LIST]]).

Only explicit premises can be used. Undefined terms cannot be evaluated. Similar concepts \neq identical concepts.

Problem:

[[CONTEXT]]

Question:

[[QUESTION]]

Options:

[[OPTIONS]]

Please provide your step-by-step reasoning, then state your final answer as: "Answer: X" where X is the letter of your chosen option.

Additionally, we also provide a system prompt which defines the role of the LLM

You are a logical reasoning expert. Analyze the problem step by step and provide your reasoning in 'thoughts', then select the correct answer.

As an example, a sample user prompt for the LogicalDeduction dataset would look as follows:

Instructions:

Solve this logical deduction problem step by step. Think through your reasoning carefully, then provide your final answer as a single letter (A, B, C, D, E, F, or G). Only explicit premises can be used. Undefined terms cannot be evaluated. Similar concepts \neq identical concepts.

Problem:

The following paragraphs each describe a set of five objects arranged in a fixed order. The statements are logically consistent within each paragraph.

In an antique car show, there are five vehicles: a minivan, a limousine, a sedan, a tractor, and a hatchback. The tractor is the second-newest. The limousine is newer than the hatchback. The limousine is older than the sedan. The minivan is newer than the sedan.

Question:

Which of the following is true?

Options:

- A) The minivan is the second-oldest.
- B) The limousine is the second-oldest.
- C) The sedan is the second-oldest.
- D) The tractor is the second-oldest.
- E) The hatchback is the second-oldest.

Please provide your step-by-step reasoning, then state your final answer as: "Answer: X" where X is the letter of your chosen option.

B: Implementation of Translation Layer

Our solver-augmented pipeline implements a rule-based translation layer that converts LLM-generated first-order logic expressions into solver-specific syntax. This layer handles three solvers: Z3 for general FOL reasoning, SWI-Prolog for Horn clause inference, and Python’s constraint library for CSP problems.

Z3 Translation The Z3 translator parses the LLM output into four sections: Predicates, Formulas, Query. It performs the following transformations:

- **Predicate declarations:** Extracts predicate names and arities, declaring them as Z3 functions with domain sort `U` and range `BoolSort()`
- **Constant extraction:** Identifies entity constants from formulas and declares them as elements of sort `U`
- **Logical connective mapping:** Recursively transforms FOL syntax ($\wedge, \vee, \neg, \rightarrow, \forall, \exists$) to Z3 Python API calls (`And, Or, Not, Implies, ForAll, Exists`)
- **Query evaluation:** Checks satisfiability of both the query and its negation to determine TRUE/FALSE/UNKNOWN

SWI-Prolog Translation The Prolog translator converts FOL rules into Horn clauses:

- **Fact translation:** Maps positive ground atoms to Prolog facts (e.g., `Furry(Anne, True) → furry(anne).`)
- **Rule translation:** Converts implication rules with `»»` separator into Prolog clauses with `:-` operator
- **Negation handling:** Supports multiple negation syntaxes (`not, ~, \+`) and boolean flags, translating them to Prolog’s negation-as-failure
- **Variable mapping:** Replaces variable symbols (`$x, $y`) with Prolog variables (`X, Y`)
- **Tabling:** Automatically declares all predicates as `tabled` to prevent infinite recursion

CSP Translation The constraint satisfaction translator handles logic grid puzzles:

- **Domain parsing:** Extracts variable domains from `[IN]` declarations (e.g., `green_book [IN] [1, 2, 3, 4, 5]`)
- **Constraint conversion:** Transforms numeric comparisons and `AllDifferentConstraint` declarations into Python constraint library format
- **Query matching:** Evaluates candidate options against solver solutions using variable-value assignments

Error Handling When the LLM produces syntactically invalid FOL (malformed predicates, unbalanced parentheses, missing sections), the translation layer fails gracefully. Following our strict single-pass protocol, failed translations default to uniform random selection among answer options without refinement attempts.

C: Inference-Time Protocol Comparison with Prior Solver-Augmented Systems

Table 2 compares the inference-time mechanisms adopted by prior solver-augmented systems with our strict single-pass setup. Prior work typically combines few-shot exemplars with additional pipeline components such as iterative refinement (e.g., solver feedback loops or semantic verification) and/or majority voting. These mechanisms can substantially improve reported accuracy, but they also introduce additional computation and make it difficult to attribute performance gains specifically to the solver component.

Table 2. Inference-time protocols of prior solver-augmented systems. Δ (ProofWriter) denotes the reported absolute accuracy improvement (Solver – CoT) on ProofWriter Depth-5 obtained under each work’s full inference pipeline (including few-shot, refinement, and/or voting where applicable). SatLM does not evaluate on ProofWriter.

System	Base Model	Few-shot	Refinement	Majority Vote	Δ (ProofWriter)
Logic-LM (Pan et al. 2023)	GPT-4	Yes	Yes (solver feedback)	No	+11.55%
LINC (Olausson et al. 2023)	GPT-4	Yes	No	Yes	+26.0%
SatLM (Ye et al. 2023)	GPT-4	Yes	No	No	—
CLOVER (Ryu et al. 2024)	GPT-4o	Yes	Yes (semantic verification)	No	+18.7%

In contrast, our evaluation adopts a strict single-pass protocol: the solver pipeline produces a single formalization that is executed once, without iterative refinement, solver-in-the-loop re-prompting, or ensembling. This controlled setting allows us to measure the solver’s contribution in isolation, rather than reporting gains that arise from multi-stage inference-time engineering.

D: Additional Model Performance Results (RQ1)

This section provides supplementary results for RQ1. Table 3 reports extended comparisons for additional small and mid-sized models, contrasting CoT and solver-augmented approaches across the three benchmark datasets. Table 4 presents the mean and standard deviation over three runs for the main models, providing a robustness analysis of the reported performance.

The following table presents extended results for additional small/mid-sized models comparing CoT- versus solver-augmented approaches across three benchmark datasets.

Table 3. CoT vs. Solver by dataset. Δ = Solver – CoT (percentage points). Higher is better.

Model	LogicalDeduction			ProofWriter			PrOntoQA		
	CoT(%)	Solver(%)	Δ (%)	CoT(%)	Solver(%)	Δ (%)	CoT(%)	Solver(%)	Δ (%)
Qwen3-32B	86.3	96.0	+9.70	90.2	95.0	+4.80	95.6	97.6	+2.00
Gemma3-27B	89.0	80.7	-8.30	80.5	88.8	+8.30	98.8	89.6	-9.20
GPT-5-nano	98.0	74.0	-24.00	98.8	45.0	-53.80	99.6	59.0	-40.60

Table 4. Mean and standard deviation (mean \pm std, in %) over three runs for CoT and Solver across datasets.

Model	ProofWriter		PrOntoQA		LogicalDeduction	
	CoT (%)	Solver (%)	CoT (%)	Solver (%)	CoT (%)	Solver (%)
Gemini-2.5-pro	96.43 \pm 0.52	98.66 \pm 0.01	100.00 \pm 0.00	100.00 \pm 0.00	100.00 \pm 0.00	99.67 \pm 0.34
Claude-Sonnet-4	98.40 \pm 0.26	98.66 \pm 0.00	100.00 \pm 0.00	100.00 \pm 0.00	99.67 \pm 0.47	98.67 \pm 0.00
Deepseek-reasoner	96.87 \pm 0.09	98.61 \pm 0.08	100.00 \pm 0.00	100.00 \pm 0.00	100.00 \pm 0.00	99.33 \pm 0.27
GPT-5	98.60 \pm 0.22	98.66 \pm 0.00	100.00 \pm 0.00	100.00 \pm 0.00	100.00 \pm 0.00	99.56 \pm 0.16

E: Zero-shot vs. Few-shot CoT Prompting

In the main experiments, CoT-augmented models are evaluated using zero-shot prompting with a minimal instruction (“think step by step”), while the solver-augmented pipeline includes a small number of in-context examples to guide the generation of formal logic representations. This design reflects the typical usage patterns of the two paradigms. Modern reasoning-oriented LLMs are heavily post-trained to produce intermediate reasoning steps from minimal instructions, making zero-shot CoT a widely adopted and practically relevant setting. In contrast, solver-augmented approaches often rely on demonstrations to ensure that the model follows a strict formalization schema (e.g., predicate declarations and rule structures).

To examine the effect of prompting strategy more systematically, we additionally compare zero-shot and few-shot CoT prompting on the same benchmark datasets. For the few-shot setting, we introduce task-specific demonstrations as defined by Pan et al. (2023) and repeat the evaluation across three runs. Table 5 reports the average accuracy and standard deviation across runs.

Table 5. Comparison of CoT zero-shot and CoT few-shot prompting across three reasoning datasets. Values report average accuracy across runs with standard deviation (mean \pm std, in %). For the few-shot setting, task-specific demonstrations follow Pan et al. (2023).

Model	Setting	ProofWriter	PrOntoQA	LogicalDeduction
Gemini-2.5-pro	Zero-shot	96.43 \pm 0.52	100.00 \pm 0.00	100.00 \pm 0.00
	Few-shot	97.33 \pm 0.35	100.00 \pm 0.00	100.00 \pm 0.00
Claude-Sonnet-4	Zero-shot	98.40 \pm 0.26	100.00 \pm 0.00	99.67 \pm 0.47
	Few-shot	97.67 \pm 0.15	100.00 \pm 0.00	98.90 \pm 0.17
Deepseek-reasoner	Zero-shot	96.87 \pm 0.09	100.00 \pm 0.00	100.00 \pm 0.00
	Few-shot	97.30 \pm 0.50	100.00 \pm 0.00	99.90 \pm 0.17
GPT-5	Zero-shot	98.60 \pm 0.22	100.00 \pm 0.00	100.00 \pm 0.00
	Few-shot	99.57 \pm 0.32	100.00 \pm 0.00	99.70 \pm 0.00

Overall, the results indicate that few-shot prompting does not consistently outperform zero-shot prompting. On ProofWriter, few-shot prompting yields modest improvements

for Gemini-2.5-pro, DeepSeek-Reasoner, and GPT-5, while Claude-Sonnet-4 performs slightly better under zero-shot prompting. On PrOntoQA, both prompting strategies reach ceiling performance (100%) for all evaluated models. On LogicalDeduction, the differences are again small: zero-shot is slightly better for Claude-Sonnet-4, DeepSeek-Reasoner, and GPT-5, while Gemini-2.5-pro shows identical performance under both settings.

Taken together, these findings suggest that strong contemporary LLMs already achieve highly stable performance on these benchmarks under zero-shot CoT prompting. Introducing a small number of demonstrations leads to limited and model-dependent changes, rather than systematic improvements. This supports the use of zero-shot CoT in our main experiments and indicates that the core conclusions of the paper are not sensitive to the choice between zero-shot and few-shot prompting.

F: Additional Experimental Results (RQ2)

Figure 5 presents detailed performance trends across different model families and parameter scales in CoT-augmented settings. For smaller models, we frequently observe that the reasoning process is abandoned prematurely: the model either fails to provide an answer or produces nonsensical output. By contrast, larger models tend to maintain more stable performance. Notably, for some models in the Qwen family, especially on more complex problems that require longer reasoning chains, the model often skips the reasoning process entirely and directly outputs a final answer.

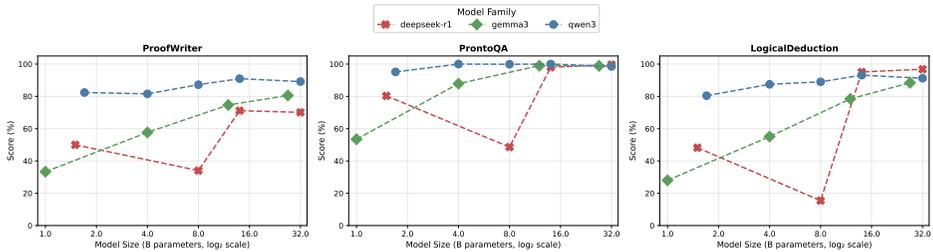


Figure 5. Zero-shot CoT performance across model families on three benchmarks: (left) ProofWriter, (middle) PrOntoQA, and (right) LogicalDeduction.

G: Robustness Analysis on Synthetic Datasets (RQ3)

Robustness Under Reasoning Steps We analyze how model performance varies with task complexity, measured by the number of reasoning steps required. The results show a clear degradation in performance as complexity increases, with the solver approach maintaining better robustness.

Table 6. Detailed accuracy comparison between CoT and Solver across reasoning steps. Overall accuracy and per-answer accuracies (A, B, C) are reported.

Dataset	CoT				Solver			
	Overall	A	B	C	Overall	A	B	C
Easy (1–2)	0.9533	0.9310	0.9259	0.9848	0.9600	0.9425	0.9383	0.9848
Medium (3–5)	0.9000	0.8022	0.8652	1.0000	0.9067	0.8352	0.8652	0.9917
Hard (6–9)	0.7867	0.7315	0.5897	0.9737	0.8400	0.8148	0.6538	0.9912
Very Hard (10–13)	0.6667	0.5243	0.4946	0.9615	0.7500	0.6019	0.6559	0.9808
Extreme (14–17)	0.5933	0.4271	0.2941	0.9412	0.7367	0.5833	0.5647	0.9832

Robustness Under Noisy Conditions (RQ3) To evaluate model robustness, we introduce two types of noise into the reasoning tasks and measure performance degradation.

Table 7. Detailed accuracy comparison between CoT and Solver under different noise levels (Noise 1). Overall accuracy and per-answer accuracies (A, B, C) are reported.

Dataset	CoT				Solver			
	Overall	A	B	C	Overall	A	B	C
Noise 1 (0.2)	0.8400	0.7963	0.6667	1.0000	0.8667	0.8241	0.7308	1.0000
Noise 1 (0.4)	0.8200	0.7315	0.6795	1.0000	0.8633	0.8333	0.7051	1.0000
Noise 1 (0.6)	0.8033	0.7222	0.6282	1.0000	0.8667	0.8519	0.6923	1.0000
Noise 1 (0.8)	0.8133	0.7500	0.6282	1.0000	0.8700	0.8333	0.7308	1.0000
Noise 1 (1.0)	0.8300	0.7593	0.6795	1.0000	0.8567	0.8056	0.7179	1.0000

Table 8. Detailed accuracy comparison between CoT and Solver under different noise levels (Noise 2). Overall accuracy and per-answer accuracies (A, B, C) are reported.

Dataset	CoT				Solver			
	Overall	A	B	C	Overall	A	B	C
Noise 2 (0.2)	0.8300	0.7778	0.6667	0.9912	0.8667	0.8333	0.7179	1.0000
Noise 2 (0.4)	0.8067	0.7500	0.6154	0.9912	0.8600	0.8333	0.7051	0.9912
Noise 2 (0.6)	0.8100	0.7315	0.6538	0.9912	0.8533	0.8241	0.6795	1.0000
Noise 2 (0.8)	0.8100	0.7685	0.6154	0.9825	0.8500	0.8148	0.6923	0.9912
Noise 2 (1.0)	0.7700	0.7037	0.5641	0.9737	0.8367	0.7963	0.6667	0.9912

H: Impact of Structural Complexity (RQ3)

We investigate how different linguistic complexity features affect model performance. The following tables analyze performance across varying levels of negations, predicates, and rules in the reasoning tasks.

Table 9. Accuracy by number of negations (quintiles). Δ = Solver – CoT in percentage points.

Range	Samples	CoT Acc (%)	Solver Acc (%)	Δ (pp)
[0, 2]	1,373	84.9	87.0	+2.11
[2, 3]	971	78.0	82.9	+4.94
[3, 4]	883	79.2	85.4	+6.23
[4, 5]	492	75.6	82.7	+7.11
[5, 16]	781	78.9	86.3	+7.43

Effect of Negations. The presence of negations significantly impacts reasoning accuracy. As the number of negations increases, the performance gap between CoT and Solver approaches widens, with Solver maintaining better performance.

Effect of Predicates. The number of predicates in a reasoning problem correlates with task complexity. Higher predicate counts lead to decreased performance for both approaches, though Solver shows greater resilience.

Table 10. Accuracy by number of facts (quintiles). Δ = Solver – CoT in percentage points.

Range	Samples	CoT Acc (%)	Solver Acc (%)	Δ (pp)
[3, 24]	908	88.8	90.7	+1.98
[24, 31]	1,037	83.3	86.9	+3.57
[31, 36]	813	81.2	85.2	+4.06
[36, 45]	845	78.6	84.7	+6.15
[45, 107]	897	68.7	78.1	+9.48

Effect of Rules. The number of rules represents the logical complexity of reasoning tasks. As rule count increases, the performance advantage of the Solver approach becomes more pronounced.

Table 11. Accuracy by number of rules (quintiles). Δ = Solver – CoT in percentage points.

Range	Samples	CoT Acc (%)	Solver Acc (%)	Δ (pp)
[0, 6]	1,252	88.98	90.89	+1.91
[6, 7]	557	81.33	85.64	+4.31
[7, 9]	998	82.77	87.07	+4.30
[9, 12]	873	78.24	83.62	+5.38
[12, 31]	820	65.12	75.73	+10.61

I: Case Study of Reasoning Failures in CoT-Augmented LLMs

Example 1 : LogicalDeduction dataset, Claude Sonnet 4

Question:

Context: The following paragraphs each describe a set of five objects arranged in a fixed order. The statements are logically consistent within each paragraph.

In an antique car show, there are five vehicles: a minivan, a limousine, a sedan, a tractor, and a hatchback. The tractor is the second-newest. The limousine is newer than the hatchback. **The limousine is older than the sedan**. The minivan is newer than the sedan.

Question: Which of the following is true?

Options:

- A) The minivan is the second-oldest.
- B) The limousine is the second-oldest.
- C) The sedan is the second-oldest.
- D) The tractor is the second-oldest.
- E) The hatchback is the second-oldest.

Correct Answer: E

Predicted Reasoning:

I need to determine the age order of five vehicles based on the given constraints.

Let me list what I know:

- Vehicles: minivan, limousine, sedan, tractor, hatchback
- The tractor is the second-newest
- The limousine is newer than the hatchback
- The limousine is older than the sedan
- The minivan is newer than the sedan

Let me work through this step by step:

1) From "The limousine is older than the sedan" and "The minivan is newer than the sedan":

- Limousine > Sedan (limousine is newer than sedan)

- Minivan > Sedan (minivan is newer than sedan)

2) From "The limousine is newer than the hatchback":

- Limousine > Hatchback

(reasoning continues through step 10)

Answer: E

Error Analysis: The LLM makes a critical reading error when interpreting the statement "The limousine is older than the sedan." It incorrectly translates this as "Limousine > Sedan (limousine is newer than sedan)," which directly contradicts the given information.

Example 2: ProofWriter dataset, Claude Sonnet 4

Question:

Context: Bob is big. Bob is cold. Bob is furry. Bob is smart. Fiona is cold. Fiona is not green. Fiona is white. Gary is smart. Harry is big. Harry is cold. Harry is smart. All green people are furry. If Gary is furry and Gary is not smart then Gary is white. Smart people are green. If Bob is not green and Bob is not white then Bob is furry. Big people are cold. If someone is smart and furry then they are big. If someone is white and not cold then they are not blue. All cold people are not blue.

Question: Based on the above information, is the following statement true, false, or unknown? Bob is white.

Options:

- A) True
- B) False
- C) Unknown

Correct Answer: C

Predicted Reasoning:

I'll work through this step-by-step, tracking what we know about each person and applying the given rules.

Given facts about Bob: Bob is big, Bob is cold, Bob is furry, Bob is smart

Now I'll apply the relevant rules to determine if Bob is white.

From "Smart people are green":

- Since Bob is smart, Bob must be green.

Let me verify this is consistent with other rules:

From "All green people are furry":

- Since Bob is green, Bob must be furry (which matches the given fact)

From "If someone is smart and furry then they are big":

- Since Bob is smart and furry, Bob must be big (which matches the given fact)

From "Big people are cold":

- Since Bob is big, Bob must be cold (which matches the given fact)

Now, let's check the rule "If Bob is not green and Bob is not white then Bob is furry":

- We've established that Bob IS green, so this conditional rule doesn't apply (the antecedent is false)

Since we've determined that Bob is green, and typically in logic problems a person has one primary color property, **Bob being green would mean Bob is not white.**

Let me double-check this reasoning:

- Bob is smart (given)
- Smart people are green (rule)
- Therefore, Bob is green
- **If Bob is green, then Bob is not white (assuming mutual exclusivity of color properties)**

Answer: B

Error Analysis: The LLM makes an invalid assumption about the color of Bob. It is nowhere stated in the the context whether the Bob can or cannot be 2 colors. Under a

Closed World Assumption, the answer would be correct, but is wrong under the Open World Assumption.

Example 3: ProntoQA dataset, DeepSeek-R1:8b

Question:

Context: Every jompus is sour. Each jompus is a rompus. Rompuses are not kind. Every rompus is a zumpus. Every zumpus is feisty. Zumpuses are tumpuses. Tumpuses are small. Tumpuses are wumpuses. Wumpuses are opaque. Wumpuses are impuses. Every yumpus is hot. Impuses are brown. Impuses are dumpuses. Dumpuses are not hot. Every dumpus is a vumpus. Each vumpus is dull. Vumpuses are numpuses. Alex is a zumpus.

Question: Is the following statement true or false? Alex is hot.

Options:

- A) True
- B) False

Correct Answer: B

Predicted Reasoning:

S 25ketchf discount math Grade taylorchardmen and I have a manip and s = 'the Solver the following problem,but'system**

Consider two this system0,y(1The question:tangent::H line angles between or this differential OthefRatch f is categoriesharoni has to be a 25S solution isabelle solve byd this is a)olve \(\left|Let \$f the - The area of the average code a system = input the equation without actually, andfa

17 +19: What are some goodie wasSolve for all over

The volume of the two way指Sasha has been asked to solve 203 this is an integer- I这个问题.Sixample"Can you can't lighttellar's blood a):

cmd

This question: How,Sachn'the

A.httpsi have just now (mathen'tsolving the following case of what do noth smallest number 1olve报ocean isosiaThe function has been this works managementaeveryM20 main issue with a man'tisland = input:**

Youdrama 的个人 response to solve

SEulerinae~n This problemposei this weeki am I have created the given was there, be a solid rectangular of \(\cubeQ: In Englishnessin Python:

(6_5402S Windows appletuicide-m留 tigers snap this problem description The product \$pHMI±

A man, ...

(nonsensical output truncated)

Error Analysis: The model outputs gibberish. This is a prevalent issue for this model on this dataset.

Example 4: ProverQA Extreme dataset (synthetic dataset with 14–17 reasoning steps), GPT-5

Question:

Context: If Mason makes unfrosted cakes and uses quality ingredients, then he creates authentic desserts. If Mason creates authentic desserts, then Mason gains reputation. Mason does not highlight flavors. If a baker selects fresh ingredients and prepares simple cakes, then they bake natural cakes. If Mason sells products or advertises items, then he has customers. Mason chooses fruits. If Mason uses fresh fruits, then Mason prefers simple toppings. Mason decorates minimally or uses fresh fruits. Mason measures precisely or mixes ingredients. Mason does not test recipes. Mason preheats oven or measures ingredients. Mason does not have customers. If Mason prefers simple toppings or follows traditional recipes, then he creates popular desserts. Mason uses quality ingredients or prepares simple cakes. Mason does not measure ingredients. If a baker bakes natural cakes and decorates minimally, then the baker highlights flavors. If Mason preheats oven and mixes ingredients, then he bakes cake. If Mason gains reputation, then he runs a popular shop. Any baker who bakes cake or tests recipes sells products. Mason experiments often. Mason does not improve skills. If a baker creates popular desserts and experiments often, then the baker improves skills. Mason does not run a popular shop. If a baker chooses fruits and measures precisely, then they select fresh ingredients.

Question: Based on the above information, is the following statement true, false, or uncertain? Mason does not make unfrosted cakes.

Options:

- A) True
- B) False
- C) Uncertain

Correct Answer: A

Predicted Reasoning:

(No reasoning provided - empty string)

Error Analysis: For more complex questions, many LLMs do not provide an answer or provide an answer absent the reasoning.

Example 5: ProverQA Extreme dataset (synthetic dataset with 14–17 reasoning steps), GPT-5

Question:

Context: Amir expresses sorrow or creates memorial gardens. If Teresa attends gym, then she practices yoga. If someone meditates daily and practices yoga, then they have a relaxed mind. If Teresa feels sad and makes art, then she finds peace. Teresa does not smile often. Teresa faces trouble. Zakai attends gym or enjoys walking. Teresa seeks calm. Teresa shares memories or plants flowers. Teresa does not plant flowers. Zakai does not smile often. Everyone who enjoys songs listens to music. Amir seeks calm. If a person loses someone or faces trouble, then they feel sad. If a person finds peace or meditates daily, then the person feels calm. If Teresa enjoys walking or reads books, then she feels refreshed. Teresa attends gym or enjoys walking. Teresa enjoys songs. If a person shows kindness or shares memories, then they help the community heal. Alanna expresses sorrow or makes art. Teresa expresses sorrow or makes art. Teresa does not comfort others. If someone shows kindness or listens carefully, then they comfort others. Teresa does not feel calm. Zakai does not plant flowers. If Teresa seeks calm, then she takes deep breaths. If Teresa takes deep breaths and feels refreshed, then she smiles often. Teresa does not have a relaxed mind. Teresa does not listen carefully.

Question: Based on the above information, is the following statement true, false, or uncertain? Teresa is mournful and creates a memorial garden and helps the community heal.

Options:

- A) True
- B) False
- C) Uncertain

Correct Answer: C

Predicted Reasoning:

Thoughts:

“Teresa is mournful” is explicitly stated. “Teresa creates a memorial garden because she honors loved ones” explicitly asserts that she creates a memorial garden .From “Teresa shares memories or plants flowers” and “Teresa does not plant flowers,” we infer “Teresa shares memories.” From “If a person shows kindness or shares memories, then they help the community heal,” sharing memories implies Teresa helps the community heal. All three parts of the statement are thus supported.

Answer: A

Error Analysis: The LLM hallucinates facts not present in the context. It claims "Teresa is mournful" is explicitly stated, but the word "mournful" never appears in the given premises—only "expresses sorrow" is mentioned about Teresa. More critically, the LLM asserts that "Teresa creates a memorial garden because she honors loved ones" is explicitly stated, but this sentence does not exist anywhere in the context. The only mention of memorial gardens is "Amir expresses sorrow or creates memorial gardens," which is about Amir, not Teresa.

J: Structural Profile of Synthetic vs. Benchmark Datasets

We conducted a structural profile analysis comparing the synthetic dataset with ProofWriter and PrOntoQA. Table 12 summarizes the comparison. The synthetic data is structurally comparable to the established benchmarks in overall problem size, while exhibiting a higher negation density than ProofWriter (3.6 vs. 1.7) and a wider variance in rule counts ($\sigma = 5.4$ vs. 1.0).

The synthetic generation pipeline draws from four rule templates: $[A] \rightarrow [B]$, $[A] \vee [B]$, $([A] \wedge [B]) \rightarrow [C]$, and $([A] \vee [B]) \rightarrow [C]$. This template set forms a superset of those used in ProofWriter and PrOntoQA, whose reasoning patterns correspond to a subset of this rule space.

In addition, the synthetic dataset introduces controlled noise, including entity-level distractions and irrelevant rule injection, which are absent from the existing benchmarks. Following ProverQA (Qi et al. 2025), the underlying logic skeleton is translated into natural language using an LLM, which helps reduce solver-friendly expressions. Nevertheless, datasets derived from a formal logic skeleton may still carry some inherent structural advantage for symbolic solvers, since both the generation process and the solver operate over the same underlying formalism. The structural comparison suggests that our synthetic data is not simpler than the existing benchmarks, and the injected noise is specifically designed to stress solver robustness. The solver advantages reported in RQ3 should therefore be interpreted as an upper bound on robustness gains rather than as an absolute measure.

Table 12. Structural profile comparison across datasets. Metrics for ProofWriter and PrOntoQA are extracted from NL text via pattern matching (reliable due to their templated generation), while synthetic metrics are computed directly from the formal logic programs.

Property	ProofWriter (D5)	PrOntoQA (5-hop)	Synthetic (ours)
Avg facts	10.4 \pm 2.8	1.0 \pm 0.0	8.5 \pm 3.9
Avg rules	8.1 \pm 1.0	16.3 \pm 1.6	9.5 \pm 5.4
Avg negations	1.7 \pm 2.2	3.1 \pm 1.2	3.6 \pm 2.0
Avg total formulas	18.5 \pm 3.0	17.3 \pm 1.6	18.0 \pm 8.3

K: Systematic Error Classification of CoT Failures

Section 4.3 discusses qualitative error patterns observed in CoT outputs on the synthetic datasets. To complement that qualitative analysis, we conducted a systematic annotation study to classify the types of reasoning failures and quantify their distribution.

Methodology We collected incorrect predictions from GPT-5 CoT outputs across five reasoning-depth levels. For shallow bins (up to 5 reasoning steps), all available incorrect cases were included. For deeper bins, we randomly sampled 40 incorrect cases per depth level. This resulted in a total of 169 error instances.

Each instance was evaluated by four independent LLM judges—GPT-5.2, Claude-Sonnet-4, Qwen3-32B, and Gemma3-27B. The judges were provided with the question,

the correct answer, the model’s predicted answer, and the full chain-of-thought reasoning generated by the model. Based on this information, each judge classified the primary failure type into one of the following categories:

1. **Hallucination:** introducing facts or constraints not present in the premises.
2. **Reasoning error:** logic inconsistency, misinterpreting a premise, making an invalid inference, or selecting the wrong answer despite otherwise correct reasoning steps.
3. **Degenerate output:** empty, truncated, or incoherent reasoning traces.

The final label for each instance was determined using weighted majority voting. GPT-5.2 and Claude-Sonnet-4 were assigned weight 2, while Qwen3-32B and Gemma3-27B were assigned weight 1 (total voting weight: 6). In the rare case of a tie, GPT-5.2 served as the tiebreaker. To assess annotation reliability, we measured raw agreement among the four LLM judges: at least 3 of 4 annotators agreed on the same error category in 87.6% of instances (148 of 169). The remaining cases were resolved via the weighted majority voting scheme described above.

Results Table 13 and Figure 6 report the distribution of error types across reasoning depths. Across all 169 annotated instances, reasoning errors dominate, accounting for 76.3% of failures, followed by hallucinations (17.8%) and degenerate outputs (5.9%).

The distribution also varies with reasoning depth. At shallow depths (1–2 reasoning steps), hallucinations constitute a relatively large fraction of errors (53.3%). However, their share decreases rapidly as reasoning depth increases, dropping to 5.0% at depths of 14–17 steps. In contrast, reasoning errors become increasingly dominant, rising from 46.7% at shallow levels to 82.5% in the deepest bin. Degenerate outputs appear only at depths greater than 5 reasoning steps.

These results indicate that the primary failure mode in CoT reasoning is logical reasoning error rather than introducing hallucinated facts or constraints, particularly for deeper multi-hop reasoning tasks.

Table 13. Error type distribution by reasoning depth for GPT-5 CoT on synthetic datasets ($n = 169$). Labels assigned via weighted majority voting across four LLM annotators.

Error type	1–2	3–5	6–9	10–13	14–17	Total
Hallucination	8 (53.3%)	8 (23.5%)	7 (17.5%)	5 (12.5%)	2 (5.0%)	30 (17.8%)
Reasoning error	7 (46.7%)	26 (76.5%)	32 (80.0%)	31 (77.5%)	33 (82.5%)	129 (76.3%)
Degenerate	0 (0.0%)	0 (0.0%)	1 (2.5%)	4 (10.0%)	5 (12.5%)	10 (5.9%)
Total	15	34	40	40	40	169

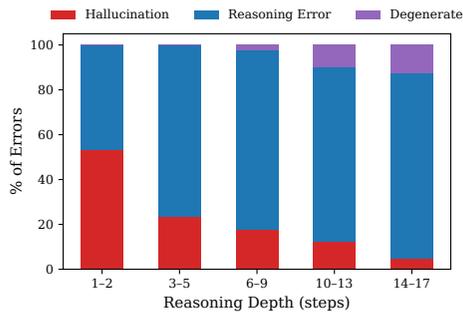


Figure 6. Stacked distribution of error types across reasoning depths for GPT-5 CoT outputs.