TOWARDS SYMBOLIC TOOL-INTEGRATED LOGICAL REASONING IN LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Logical reasoning requires deriving valid conclusions from given facts and rules and serves as a foundation for reliable decision-making. Recent advances in large language models (LLMs) have improved their reasoning capabilities, but their reliability under complex logical constraints remains insufficiently evaluated. Existing benchmarks typically involve only a few entities and clues, leaving complex reasoning underexplored. To bridge this gap, we introduce **LOGICEVAL-$\mathbb{X}$**, a benchmark of 944 expert-verified multiple-choice problems, averaging 8.31 entities, 12.47 attributes, and 18.39 clues per problem. Evaluating 10 representative LLMs and reasoning approaches shows that complex logical reasoning remains challenging, with the best-performing model, GPT-5.5, achieving only 52.65% accuracy. We further identify ***Lost in Complexity***, a failure mode in which hypothesis enumeration becomes prolonged and unreliable as complexity increases. To address this issue, we propose Tool-Integrated Monte Carlo Tree Search ($\mathbb{T}$-MCTS), a post-training framework that trains LLMs to formulate problems symbolically and delegate complex inference to external solvers. By exploring symbolic formulations and solver interactions, $\mathbb{T}$-MCTS generates solver-verified trajectories and process-level rewards for autonomous tool use and feedback-driven revision. Based on $\mathbb{T}$-MCTS, we develop LOGIC-$\mathbb{X}$-8B and LOGIC-$\mathbb{X}$-14B, achieving 53.60% and 62.71% accuracy on LOGICEVAL-$\mathbb{X}$, respectively, with LOGIC-$\mathbb{X}$-14B outperforming all evaluated methods. Further experiments show that this capability generalizes to other logical reasoning tasks. Dataset and code are available at the anonymous repository.

1 INTRODUCTION

Logical reasoning derives valid conclusions from given facts or premises through inference (Thatikonda et al., 2025). Unlike intuitive or heuristic reasoning, its conclusions follow from identifiable facts and rules, allowing the reasoning process to be traced and providing a transparent basis for reliable decision-making (Liu et al., 2025). Generative models, particularly large language models (LLMs), have advanced rapidly in recent years (Wei et al., 2022, Singh et al., 2025). Their improving language understanding and multi-step reasoning capabilities enable them to tackle increasingly complex tasks (Kojima et al., 2022). Accordingly, more studies use LLMs to automate logical reasoning (Xu et al., 2025, Wang et al., 2026, Lin et al., 2025b, Liu et al., 2025).

To assess LLMs' logical reasoning abilities, researchers have developed benchmarks covering a range of reasoning tasks (Saparov & He, 2022, bench authors, 2023, Han et al., 2024, Parmar et al., 2024). Yet recent studies suggest that many existing benchmarks have become too simple to probe the limits of these abilities (Kazemi et al., 2025, Zhang et al., 2026). Real-world decisions, such as applying enterprise rules, often require reasoning across interdependent entities, attributes, and constraints. In contrast, mainstream benchmarks typically involve only a handful of entities and stated clues, as shown in Table 1. This gap calls for a benchmark that tests whether LLMs can reason reliably under complex rules.

To fill this gap, we introduce **LOGICEVAL-$\mathbb{X}$**, a benchmark for complex logical reasoning. It comprises 944 multiple-choice problems drawn from real-world sources and curated through human difficulty screening, logic-preserving rewriting, and expert verification. LOGICEVAL-$\mathbb{X}$ exhibits three key characteristics: (1) higher *Entity Count*, averaging 8.31 entities per problem; (2) more

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Attribute Slots, with an average of 12.47 unknown attributes to resolve per problem; and (3) more *Stated Clues*, with an average of 18.39 clues given in each problem statement.

We evaluate 10 representative LLMs on LOGICEVAL-𝕏. Complex logical reasoning remains challenging: even GPT-5.5, the best-performing model, achieves only 52.65% accuracy, while most models score below 40%. Our analysis reveals a recurring strategy of hypothesis enumeration, in which models test candidate solutions one by one. Although effective on simpler problems, this strategy leads to prolonged reasoning as complexity increases. During enumeration, models lose track of stated constraints (38%) or introduce unsupported ones (18%). We term this failure mode *Lost in Complexity*.

Table 1: Comparison of logical reasoning benchmarks, where *Entity Count*, *Attribute Slots*, and *Stated Clues* are the average numbers of entities, unknown attributes, and clues across sampled problems, respectively.

Benchmark	Num.	Entity Count	Attribute Slots	Stated Clues
ProofWriter (Tafjord et al., 2021)	600	2.27	1.04	8.16
BIG-bench (bench authors, 2023)	1,270	4.43	5.36	5.42
AR-LSAT (Zhong et al., 2021)	230	4.17	6.53	5.38
FOLIO (Han et al., 2024)	204	1.24	1.59	5.31
ProntoQA (Saparov & He, 2022)	500	1.03	0.98	7.17
LOGICEVAL-𝕏 (Ours)	**944**	**8.31**	**12.47**	**18.39**

Lost in Complexity exposes the burden of hypothesis enumeration: LLMs must track interdependent constraints while searching through candidate solutions in long reasoning traces. These constraints can be formalized and maintained by symbolic solvers. We therefore study *Symbolic Tool-integrated Reasoning*, in which LLMs formulate problems symbolically and use solvers to explore solution spaces rather than enumerating hypotheses in text. As shown in Figure 5, symbolic tools improve GPT-5.5 accuracy from 52.65% to 59.64%, but do not benefit every model: Qwen3-8B drops from 26.44% to 10.34%. Further analysis identifies symbolic formulation, solver selection, and feedback-based error recovery as persistent difficulties.

To address these difficulties, we propose Tool-Integrated Monte Carlo Tree Search (𝕋-MCTS), a post-training strategy that teaches LLMs to solve logical problems symbolically. Unlike conventional MCTS (Zhang et al., 2024), which typically evaluates sampled paths through terminal outcomes, 𝕋-MCTS uses both intermediate solver feedback and final-answer correctness to guide the search over tool-use trajectories. This search yields solver-verified trajectories for supervised fine-tuning and process-level rewards for reinforcement learning. These training signals enable LLMs to use symbolic tools autonomously and adapt their reasoning when solver feedback calls for revision.

Based on 𝕋-MCTS, we develop two symbolic tool-integrated reasoning models, LOGIC-𝕏-8B and LOGIC-𝕏-14B. Experiments demonstrate that 𝕋-MCTS enables reliable symbolic tool use for complex logical reasoning, improving accuracy from 10.34% to 53.60% for LOGIC-𝕏-8B and from 17.05% to 62.71% for LOGIC-𝕏-14B, surpassing GPT-5.5 with 52.65% accuracy on LOGICEVAL-𝕏. Both models also outperform the compared reasoning methods at their respective parameter scales. Further analysis shows that 𝕋-MCTS effectively improves core capabilities of symbolic tool-integrated reasoning, including symbolic formulation, solver selection, and especially feedback-based revision. Moreover, 𝕋-MCTS generalizes across diverse logical reasoning tasks, with LOGIC-𝕏-14B improving accuracy from 51.0% to 87.8% on AR-LSAT and from 61.0% to 84.6% on ProofWriter compared with its base model. Our contributions are as follows:

- To examine LLMs on complex logical reasoning, we introduce **LOGICEVAL-𝕏**, a benchmark of 944 expert-verified problems, and uncover *Lost in Complexity*: reliance on hypothesis enumeration undermines performance as problem complexity increases.
- We propose **𝕋-MCTS**, a post-training strategy that explores symbolic formulations and solver interactions to produce solver-verified trajectories for supervised fine-tuning and process-level rewards for reinforcement learning.
- Using 𝕋-MCTS, we develop LOGIC-𝕏-8B and LOGIC-𝕏-14B. They achieve 53.60% and 62.71% accuracy on LOGICEVAL-𝕏, respectively, outperform the evaluated models and approaches, and generalize to other logical reasoning tasks.

2 RELATED WORK

Logical Reasoning Benchmarks. Existing benchmarks have established a foundation for evaluating LLMs' logical reasoning abilities. Early benchmarks study whether models can perform valid

inference over structured knowledge and natural language premises (Clark et al., 2020, Tafjord et al., 2021, Liu et al., 2020, Yu et al., 2020). Later studies extend evaluation toward more complex reasoning settings by introducing richer logical forms, diverse inference processes, and deeper reasoning chains (Han et al., 2024, Parmar et al., 2024, Patel et al., 2024). Recent benchmarks further investigate logical reasoning complexity through controllable constraints (Lin et al., 2025a, Zhang et al., 2026, Wei et al., 2025). However, as shown in Table 1, existing benchmarks offer limited insight into LLMs' ability to reason under complex constraints.

Symbolic Reasoning with LLMs. Prior work has used symbolic or executable representations to connect natural language with formal computation (Gao et al., 2023, Chen et al., 2022). Later studies integrated symbolic solvers for inference and verification (Pan et al., 2023, Kirtania et al., 2024, Olausson et al., 2023), while tool-integrated approaches extended LLM reasoning beyond text generation (Gou et al., 2024, Lin & Xu, 2025). Monte Carlo Tree Search (MCTS) has been used to explore large reasoning spaces (Hao et al., 2023), plan agent actions (Zhou et al., 2023), and collect reasoning trajectories for LLM training (Zhang et al., 2024). The use of MCTS to train LLMs for tool interaction remains underexplored, particularly for complex logical reasoning.

3 CONSTRUCTION OF LOGICEVAL-𝕏

We construct LOGICEVAL-𝕏 to evaluate complex logical reasoning. The benchmark consists of multiple-choice problems, each with a single correct answer. Its construction follows five stages: problem collection, difficulty screening, complexity annotation, rewriting, and expert verification.

Problem Collection and Quality Filtering. We collect seed problems from historical LSAT analytical reasoning questions[1], GMAT practice questions[2], mathematical logic puzzle books, constraint satisfaction puzzle collections, and discrete mathematics problem books. We use Qwen3-32B (Team, 2025) for initial quality filtering to remove problems with unclear wording, incomplete conditions, ambiguous options, or non-unique answers. Problems are converted into single-answer multiple-choice format when necessary. Further details and prompts are provided in Appendix A.1.

LOGICEVAL-X

On Gua Island there are two tribes: the Honest Tribe (always tells the truth) and the Lying Tribe (always lies). A person from the Honest Tribe married a person from the Lying Tribe, and after marriage they influenced each other:

- *The one from the Honest Tribe now tells one lie after every three truths;*
- *The one from the Lying Tribe now tells one truth after every three lies.*
- *They have a son, who alternates between truth and lie.*

Each of the three has a different tribe number, and their names are Jack, Andrew, and Wallace. Each person says 4 sentences:

A says:	**B says:**	**C says:**
1. Jack's number is the largest of the three;	...	...
2. I used to be from the Honest Tribe;		
3. B is my wife;		
4. My tribe number is 22 larger than B's.		

Question: Which of the following descriptions about the three people's identities, names, and tribe numbers is correct?

A) A is the wife, name Jack, tribe number 66; B is the husband, name Wallace, tribe number 44; C is the son, name Andrew, tribe number 54.

...

Figure 1: An example problem from LOGICEVAL-X.

Difficulty Screening. Thirteen volunteers, comprising five undergraduates and eight graduate students in science and engineering, solve the candidate problems. We rank problems by the proportion of correct responses and retain the 20% with the lowest accuracy. See Appendix A.1 for details.

Complexity Annotation. An LLM annotates the 1,023 difficulty-selected problems across three complexity dimensions. Six mathematics and statistics experts independently audit a random 10% of these problems and 10% of each comparison benchmark in Table 1. Their inter-expert agreement before resolving disagreements is ICC(2,6) = 0.80. See Appendices A.1.4 and A.1.5 for details.

Logic-preserving Rewriting. To reduce contamination, we remap each selected problem to a new theme, rewrite its statement and options, and iteratively check that its logic and answer are preserved. Details and prompts are provided in Appendix A.1.6.

Expert Verification. Experts verify each rewrite's logical consistency, reasoning structure, and answer correctness, with Fleiss' $\kappa = 0.75$. The final benchmark contains 944 verified problems. Figure 1 shows an example, with the complete problem provided in Appendix A.1.8. Verification details are provided in Appendix A.1.7.

[1] https://www.lsac.org/lsat/prepare/official-lsat-practice-tests

[2] https://www.mba.com/exams/gmat-exam/prep-for-the-exam

Table 2: Evaluation results on LOGICEVAL-X. Accuracy and Self-consistency@3 are reported as percentages (%). Output length is reported using the mean and median numbers of generated tokens.

Model	Accuracy (%) [min–max]	Output Length (μ/Med., tokens)	Self-consistency@3 (%) [min–max]
LLAMA-3.1-8B (Grattafiori et al., 2024)	26.91 [25.84–27.92]	523 / 378	38.72 [34.10–43.35]
QWEN3-8B (Team, 2025)	26.44 [25.28–27.43]	721 / 316	43.01 [38.32–48.09]
QWEN3-14B (Team, 2025)	24.58 [23.71–25.37]	428 / 245	46.53 [42.73–50.50]
QWEN3-32B (Team, 2025)	28.57 [27.49–29.50]	522 / 307	55.74 [52.78–58.71]
DEEPSEEK-V4-FLASH (Xu et al., 2026)	33.05 [32.12–33.95]	825 / 394	58.36 [53.90–62.80]
DEEPSEEK-V4-PRO (Xu et al., 2026)	31.36 [30.41–32.18]	438 / 314	64.99 [58.84–70.95]
KIMI-K2.6 (Team et al., 2026)	37.22 [35.97–38.28]	1207 / 704	62.71 [57.42–67.96]
GLM-5.2 (Zeng et al., 2026)	37.34 [36.58–38.19]	339 / 215	60.84 [56.95–64.72]
CLAUDE-SONNET-4.6 (Anthropic, 2026)	46.61 [45.33–47.78]	467 / 399	67.98 [65.16–70.74]
GPT-5.5 (OpenAI, 2026)	**52.65** [51.78–53.56]	- / -	**71.02** [68.53–73.45]

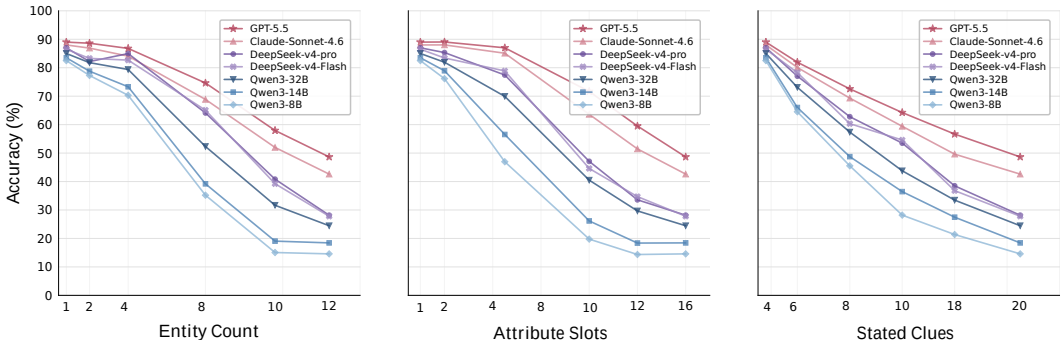

Figure 2: Accuracy trends across different dimensions of increasing problem complexity.

4 BENCHMARKING LLMS ON LOGICEVAL-𝕏

We evaluate 10 representative open-source and closed-source LLMs on LOGICEVAL-𝕏 under a unified prompting and decoding protocol, with implementation details provided in Appendix A.2.

Can Current LLMs Solve Complex Logical Reasoning Problems? As shown in Table 2, LOGICEVAL-𝕏 remains challenging for current LLMs: even the best-performing model, GPT-5.5, reaches only 52.65% accuracy, while most models remain below 40%. This suggests that scaling and stronger general reasoning capabilities alone are insufficient for reliably solving complex logical problems. Their outputs are also long, with several models exceeding 600 tokens on average and Kimi-K2.6 reaching 1207 tokens per problem, indicating that longer reasoning traces do not necessarily translate into higher accuracy. GPT-5.5 output length is not reported because its API does not expose reasoning traces. Self-consistency@3 remains limited, ranging from 38.72% to 71.02%, further showing that these failures are not merely caused by sampling instability. Overall, current LLMs still lack reliable mechanisms for complex logical reasoning under long chains, large search spaces, and dense constraints.

How Does Accuracy Vary with Problem Complexity? We analyze accuracy across *Entity Count*, *Attribute Slots*, and *Stated Clues*. To cover lower-complexity regions, we also include LLM-annotated, expert-verified samples from the benchmarks in Table 1. Figure 2 shows that accuracy declines along all three dimensions. More entities make problems harder: GPT-5.5 drops from 89.32% to 49.56% as Entity Count rises from 1 to 12. Accuracy also falls as unknown attributes accumulate: Qwen3-8B reaches only 15.16% at 16 Attribute Slots. More stated clues show the same pattern: Qwen3-14B falls below 40% at 10 Stated Clues.

Why Do LLMs Fail on Complex Logical Reasoning? To understand model failures on complex logical reasoning problems, we analyze the dominant error patterns in incorrect responses on LOGICEVAL-𝕏. As shown in Figure 3, averaged across the 10 evaluated models, Constraint Collapse, Reasoning Hallucination, and Length Exhaustion account for 38%, 18%, and 12% of incorrect responses, respectively. These failures are rarely exposed in simple reasoning tasks, but become increasingly prevalent when problems require maintaining long inference chains, tracking numerous constraints, and exploring large hypothesis spaces. Together, they reveal the *Lost in Complexity*

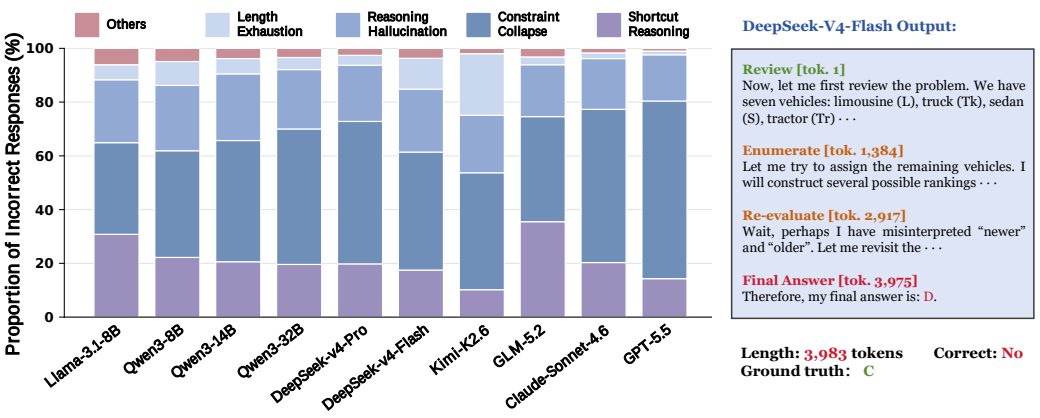

Figure 3: Distribution of model error causes on LOGICEVAL-𝕏.

Figure 4: A Lost-in-Complexity case.

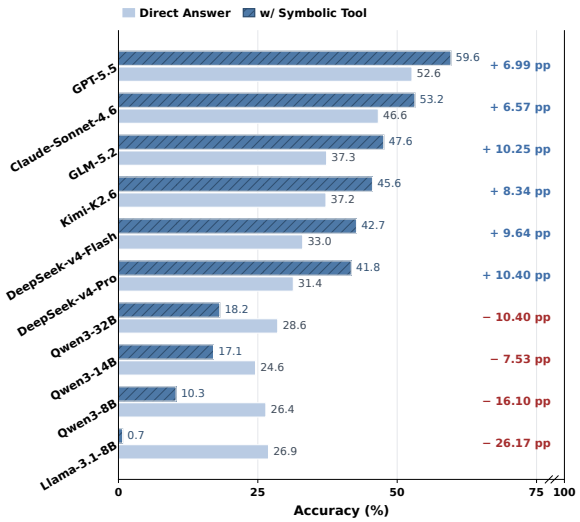

Figure 5: Preliminary study of symbolic tool-integrated reasoning on our LOGICEVAL-𝕏.

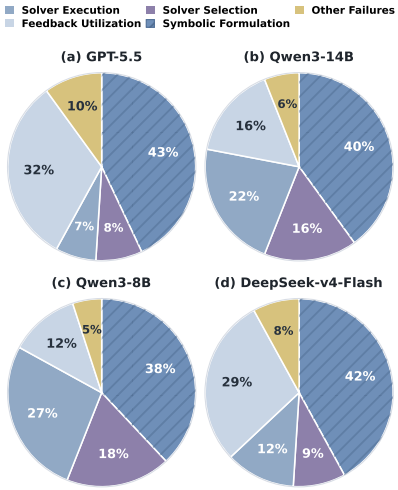

Figure 6: Distribution of the first decisive failure stages.

phenomenon: as complexity increases, LLMs tend to enumerate candidate solutions through prolonged reasoning trajectories, gradually losing constraint consistency, producing invalid intermediate deductions, and exhausting their reasoning process, as illustrated by the representative case in Figure 4.

5 PRELIMINARY STUDY OF SYMBOLIC TOOL-INTEGRATED REASONING

Can LLMs Leverage Symbolic Tools for Logical Reasoning? To examine whether symbolic tools can relieve the burden of hypothesis enumeration, we conduct a preliminary study of *Symbolic Tool-Integrated Reasoning* on LOGICEVAL-𝕏. Given a natural-language problem, an LLM constructs a symbolic representation and invokes a solver, including Pyke (Frederiksen, 2008), Prover9 (Groza, 2021), MiniZinc (Nethercote et al., 2007), or Z3 (De Moura & Bjørner, 2008), for inference. As shown in Figure 5, tool use raises GPT-5.5 accuracy from 52.7% to 59.6% and GLM-5.2 from 37.3% to 47.6%, but lowers Qwen3-8B accuracy from 26.4% to 10.3%.

Why Do LLMs Fail to Leverage Symbolic Tools? To understand these mixed results, we manually analyze 100 failed cases per model and categorize their first failure stage. As shown in Figure 6, symbolic formulation is the primary source of failure, while solver-related errors are more frequent in weaker models. Moreover, among trajectories involving an error-recovery attempt, only 18% are successfully repaired, while 82% remain unresolved: models often fail to backtrack after selecting an unsuitable solver or to correct an invalid symbolic formulation. Overall, current LLMs still struggle to use symbolic tools reliably.

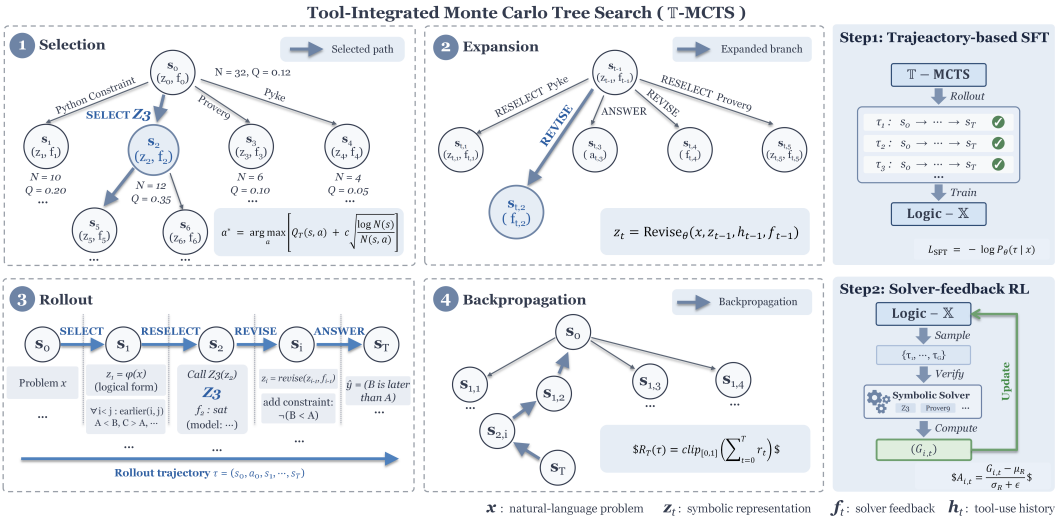

Figure 7: Overview of 𝕋-MCTS. Solver-guided search produces verified tool-use trajectories for supervised fine-tuning and process-level rewards for reinforcement learning.

6 𝕋-MCTS: Tool-Integrated Monte Carlo Tree Search

We propose Tool-Integrated Monte Carlo Tree Search (𝕋-MCTS), which uses solver-guided search to generate verified tool-use trajectories and process-level rewards for training LLMs, as illustrated in Figure 7.

Tool-use State and Actions. Given a natural-language problem x, 𝕋-MCTS defines each node as a tool-use state $s_t = (x, z_t, h_t, f_t)$, where z_t denotes the current symbolic representation, h_t records the tool-use history, and f_t represents solver feedback. Each action $a_t \in \mathcal{A}_T$ corresponds to a tool-integrated operation, including symbolic formulation, solver selection, solver invocation, feedback-based revision, or answering.

Tool-Integrated Tree Search. 𝕋-MCTS explores tool-use trajectories through four operations. When a solver is invoked, a solver-specific function ϕ_k produces executable feedback f_t from the current symbolic representation z_t, where k denotes one of the symbolic solvers introduced above.

- **Selection.** Promising tool-use actions are selected according to the upper confidence bound:

$$a^* = \arg\max_a \left(Q_T(s,a) + c\sqrt{\frac{\log N(s)}{N(s,a)}} \right), \tag{1}$$

  where $Q_T(s,a)$ denotes the estimated value of a tool-use action, $N(s)$ and $N(s,a)$ denote the state and action visit counts, and c controls exploration.
- **Expansion.** The selected state is expanded by sampling candidate symbolic formulations, solver choices, solver invocations, and reasoning continuations.
- **Tool-grounded Rollout.** From the expanded state, a tool-use trajectory is generated by invoking symbolic solvers when needed and incorporating their feedback into subsequent reasoning.
- **Solver-based Backpropagation.** The completed trajectory receives a solver-grounded return, which is backpropagated through visited nodes to update values for subsequent exploration.

For each completed trajectory $\tau = (s_0, a_0, \ldots, s_T)$, let $\mathcal{C}(\tau)$ be its solver-invocation steps and $M = \max(1, |\mathcal{C}(\tau)|)$. At each invocation, $v_t = 1$ if the symbolic program executes without error and $v_t = 0$ otherwise. We assign a step-level reward as follows:

$$r_t = \alpha \mathbb{I}[t = T] R_{\text{ans}} + \frac{\beta}{M} \mathbb{I}[t \in \mathcal{C}(\tau)] v_t - \lambda_{\text{switch}} \mathbb{I}[a_t = \textsc{Reselect}] - \lambda_{\text{repair}} \mathbb{I}[a_t = \textsc{Revise}], \tag{2}$$

Table 3: Comparison between direct answer and symbolic tool-integrated reasoning across LLMs.

Model	Setting	Accuracy (%) ↑	Output Length ↓	Self-consistency@3 (%) ↑
LLAMA-3.1-8B	Direct	26.91	523	38.72
	w/ Symbolic Tools	0.74 (-26.17)	5420 (+4897)	-
QWEN3-8B	Direct	26.44	721	43.01
	w/ Symbolic Tools	10.34 (-16.10)	4394 (+3673)	38.72 (-4.29)
QWEN3-14B	Direct	24.58	428	46.53
	w/ Symbolic Tools	17.05 (-7.53)	1266 (+838)	41.24 (-5.29)
GLM-5.2	Direct	37.34	339	60.84
	w/ Symbolic Tools	47.59 (+10.25)	1441 (+1102)	68.29 (+7.45)
CLAUDE-SONNET-4.6	Direct	46.61	467	67.98
	w/ Symbolic Tools	53.18 (+6.57)	2864 (+2397)	74.43 (+6.45)
GPT-5.5	Direct	52.65	-	71.02
	w/ Symbolic Tools	59.64 (+6.99)	-	77.51 (+6.49)
LOGIC-$\mathbb{X}$-8B (*Ours*)	Base	10.34	4394	38.72
	w/ $\mathbb{T}$-MCTS	53.60 (+43.26)	1670 (-2724)	78.43 (+39.71)
LOGIC-$\mathbb{X}$-14B (*Ours*)	Base	17.05	1266	41.24
	w/ $\mathbb{T}$-MCTS	**62.71 (+45.66)**	1665 (+399)	**84.82 (+43.58)**

and backpropagate the trajectory return $R_T(\tau) = \mathrm{clip}_{[0,1]}(\sum_{t=0}^{T} r_t)$. Here R_{ans} measures final-answer correctness; the solver-feedback term rewards execution validity at each invocation, while the remaining terms penalize solver reselection and revision.

Trajectory-based Supervised Fine-Tuning. For supervised fine-tuning, we retain only trajectories with a correct final answer and an executable final symbolic program, forming the training dataset $\mathcal{D}_{\mathbb{T}\text{-MCTS}} = \{(x_i, \tau_i)\}_{i=1}^{N}$. These trajectories serve as cold-start demonstrations for symbolic tool use. We maximize their likelihood:

$$\mathcal{L}_{\mathrm{SFT}} = -\mathbb{E}_{(x,\tau)\sim\mathcal{D}_{\mathbb{T}\text{-MCTS}}} \log P_\theta(\tau \mid x). \tag{3}$$

Solver-feedback-driven Reinforcement Learning. After trajectory-based fine-tuning, we optimize the model through online reinforcement learning. For each problem x, the model samples a group of trajectories $\{\tau_1, \ldots, \tau_G\}$. For step t of trajectory τ_i, we compute the return-to-go $G_{i,t} = \sum_{k=t}^{T_i} r_{i,k}$ and its group-normalized advantage $A_{i,t} = (G_{i,t} - \mu_R)/(\sigma_R + \epsilon)$. We then optimize individual tool-use actions:

$$\mathcal{L}_{\mathrm{RL}} = -\frac{1}{G}\sum_{i=1}^{G}\frac{1}{T_i+1}\sum_{t=0}^{T_i} A_{i,t}\log\frac{\pi_\theta(a_{i,t} \mid s_{i,t})}{\pi_{\theta_{\mathrm{old}}}(a_{i,t} \mid s_{i,t})}, \tag{4}$$

where μ_R and σ_R are computed over the step-level returns-to-go in the sampled group. Solver feedback thus updates the corresponding tool-use actions, while the final-answer reward also reaches earlier actions through the return-to-go.

Model Instantiation. We initialize LOGIC-$\mathbb{X}$-8B and LOGIC-$\mathbb{X}$-14B from Qwen3-8B-Instruct and Qwen3-14B-Instruct, respectively. Both undergo SFT on filtered $\mathbb{T}$-MCTS trajectories, followed by RL with step-level solver feedback. All 944 LOGICEVAL-$\mathbb{X}$ evaluation problems are excluded from trajectory generation and training; split and deduplication details are given in Appendix A.3.

7 EVALUATION RESULTS

$\mathbb{T}$-MCTS Enables Reliable Symbolic Tool-Integrated Reasoning. We evaluate whether LLMs can perform reliable symbolic tool-integrated reasoning on complex logical problems. As shown in Table 3, $\mathbb{T}$-MCTS substantially improves reasoning performance for both model sizes. Compared with direct symbolic tool invocation, it increases accuracy from 10.34% to 53.60% for the 8B model and from 17.05% to 62.71% for the 14B model, achieving the best performance on LOGICEVAL-$\mathbb{X}$.

Table 4: Ablation study of symbolic tool integration and training strategies on LOGICEVAL-𝕏. Changes in parentheses are relative to the Direct setting of the same model.

Model	Setting	Accuracy (%) ↑	Output Length ↓	Self-consistency@3 (%) ↑
LOGIC-𝕏-8B	Direct	26.44	721	43.01
	w/ Symbolic Tools	10.34 (-16.10)	4394 (+3673)	38.72 (-4.29)
	SFT only	42.71 (+16.27)	1850 (+1129)	61.35 (+18.34)
	SFT + Outcome RL	47.13 (+20.69)	1910 (+1189)	70.24 (+27.23)
	SFT + 𝕋-MCTS	**53.60 (+27.16)**	**1670 (+949)**	**78.43 (+35.42)**
LOGIC-𝕏-14B	Direct	24.58	428	46.53
	w/ Symbolic Tools	17.05 (-7.53)	1266 (+838)	41.24 (-5.29)
	SFT only	43.43 (+18.85)	1485 (+1057)	63.28 (+16.75)
	SFT + Outcome RL	55.72 (+31.14)	1820 (+1392)	76.15 (+29.62)
	SFT + 𝕋-MCTS	**62.71 (+38.13)**	1665 (+1237)	**84.82 (+38.29)**

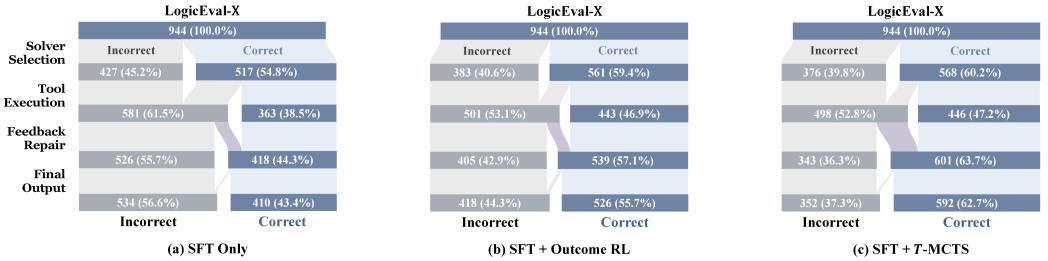

Figure 8: Stage-wise analysis of tool-use trajectories under different training strategies.

It also improves self-consistency, increasing Self-consistency@3 from 41.24% to 84.82% for the 14B model. Meanwhile, the resulting models produce outputs of 1,665–1,670 tokens. Full results are provided in Appendix A.4.

Reliable End-to-End Tool Use Is Critical for Complex Logical Reasoning. Table 5 compares LOGIC-𝕏-8B and LOGIC-𝕏-14B with reasoning-enhancement methods, agentic frameworks, and symbol-trained models. At the 8B scale, Chain-of-Thought (26.1%) offers no gain over direct reasoning (26.4%). At the 14B scale, Logic-LM++ drops from 48.6% with ground-truth solver selection to 22.6% with autonomous selection, showing that the framework struggles to select an appropriate solver for each problem. Symbol-LLM reaches 23.0% and 31.5% at the 7B and 13B scales. In contrast, LOGIC-𝕏-8B and LOGIC-𝕏-14B achieve 53.6% and 62.7%, demonstrating the effectiveness of 𝕋-MCTS in training LLMs for end-to-end symbolic tool use.

Fine-Grained Supervision Strengthens Symbolic Tool Use. Table 4 separates the contributions of 𝕋-MCTS trajectories and fine-grained rewards. Supervised fine-tuning on sampled trajectories raises accuracy from 10.34% to 42.71% for the 8B model and from 17.05% to 43.43% for the 14B model. Reinforcement learning with 𝕋-MCTS rewards adds a further 10.89 and 19.28 percentage points, respectively, compared with 4.42 and 12.29 points from outcome-only rewards. These gains show the value of both solver-verified trajectories and fine-grained reward assignment.

Solver-Grounded Rewards Improve Feedback-Based Error Recovery. We further examine the tool-use trajectories across training stages. As shown in Figure 8, 𝕋-MCTS improves solver selection, tool execution, and feedback-based repair, with its clearest advantage in repair. While SFT + Outcome RL and SFT + 𝕋-MCTS perform similarly in solver selection (59.4% vs. 60.2%) and tool execution (46.9% vs. 47.2%), 𝕋-MCTS raises the repair rate from 19.2% to 31.1% through solver-grounded trajectory rewards—an absolute gain of 11.9 percentage points and a relative improvement of 62.0%. This recovery increases accuracy from 55.7% to 62.7%, demonstrating that solver-grounded rewards enable error recovery through solver feedback.

Symbolic Tool-Integrated Reasoning Generalizes across Diverse Logical Reasoning Tasks. We examine whether symbolic tool-integrated reasoning generalizes to other logical reasoning tasks. As shown in Figure 10, both LOGIC-𝕏-8B and LOGIC-𝕏-14B consistently outperform their corre-

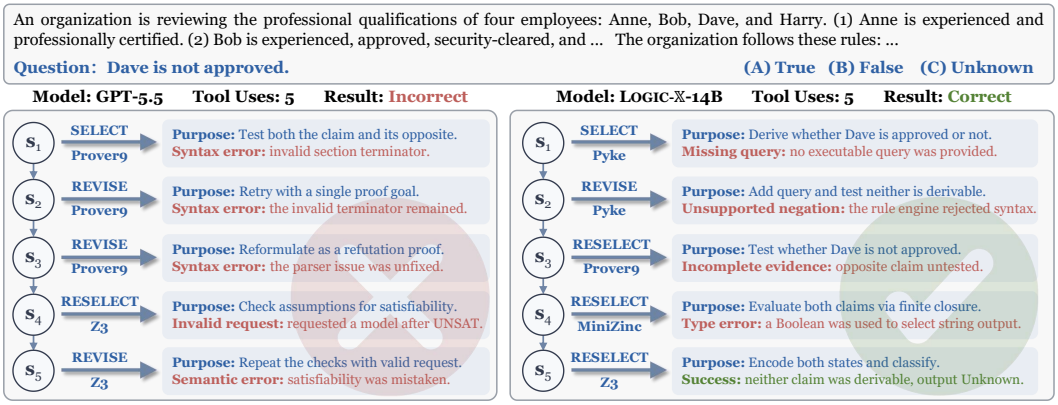

Figure 9: Case study of feedback-driven solver adaptation and semantic error recovery.

sponding Qwen3 base models across five established benchmarks. LOGIC-$\mathbb{X}$-8B improves accuracy from 24.4% to 84.4% on AR-LSAT and from 40.2% to 85.2% on ProofWriter, while LOGIC-$\mathbb{X}$-14B improves from 51.0% to 87.8% on AR-LSAT and from 61.0% to 84.6% on ProofWriter. These results demonstrate that symbolic tool-integrated reasoning transfers effectively across logical reasoning tasks with diverse problem formulations.

Case Study: Feedback-Driven Solver Adaptation. We compare GPT-5.5 and LOGIC-$\mathbb{X}$-14B on a representative problem, as shown in Figure 9. Both models invoke symbolic tools five times, but GPT-5.5 repeatedly repairs syntax errors and ultimately gives an incorrect answer by confusing satisfiability with derivability. In contrast, LOGIC-$\mathbb{X}$-14B responds to solver feedback by switching solvers, exploring alternative formulations, and correcting the semantic error to reach the correct answer, demonstrating more reliable symbolic tool-integrated reasoning.

Table 5: Comparison of reasoning methods.

Method	Params.	Accuracy (%) ↑
Small-scale Models (7B–8B)		
ReAct (Yao et al., 2022)	8B	15.6
Logic-LM++† (Kirtania et al., 2024)	8B	22.8
Symbol-LLM (Xu et al., 2024)	7B	23.0
Logic-LM† (Pan et al., 2023)	8B	23.4
Chain-of-Thought (Wei et al., 2022)	8B	26.1
LOGIC-$\mathbb{X}$-8B (*Ours*)	8B	**53.6**
Large-scale Models (13B–14B)		
ReAct (Yao et al., 2022)	14B	8.9
Chain-of-Thought (Wei et al., 2022)	14B	23.8
Symbol-LLM (Xu et al., 2024)	13B	31.5
Logic-LM† (Pan et al., 2023)	14B	48.1
Logic-LM++† (Kirtania et al., 2024)	14B	48.6
LOGIC-$\mathbb{X}$-14B (*Ours*)	14B	**62.7**

† Logic-LM and Logic-LM++ do not support autonomous solver selection and are therefore evaluated using the ground-truth solver.

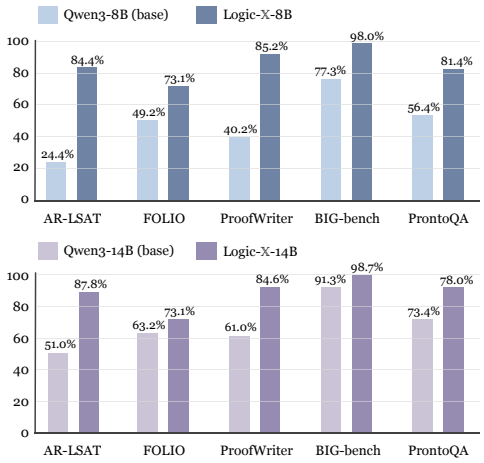

Figure 10: Comparison on established logical reasoning benchmarks.

8 CONCLUSION

We introduce **LOGICEVAL-$\mathbb{X}$**, a benchmark of 944 expert-verified problems that exposes the difficulty of complex logical reasoning for current LLMs. Across 10 evaluated models, the best accuracy is 52.65%. Our analysis identifies ***Lost in Complexity***, in which reliance on hypothesis enumeration undermines performance as complexity increases. To address this failure mode, we propose $\mathbb{T}$-MCTS, which trains LLMs to formulate problems symbolically and use solvers through verified trajectories and process-level rewards. The resulting LOGIC-$\mathbb{X}$-8B and LOGIC-$\mathbb{X}$-14B achieve 53.60% and 62.71% accuracy on LOGICEVAL-$\mathbb{X}$, respectively, and generalize to other logical reasoning tasks.

AI USE STATEMENT

In this work, we used generative AI tools to assist with identifying and refining the research topic and with drafting and editing parts of the manuscript for clarity and readability. We have not used generative AI tools for developing theoretical or conceptual frameworks, formulating mathematical claims or hypotheses, designing or providing feedback on research methodology or experiments. We have reviewed and revised all AI-assisted content and verified its accuracy. We take responsibility for the final content of this work, including text produced with the aid of generative AI.

ETHICS STATEMENT

This work focuses on logical reasoning problems and does not involve human-subject experiments. Participation in dataset screening and verification was voluntary. All annotators were compensated at hourly rates above the prevailing local rate. We are not aware of additional ethical concerns arising from this work.

REPRODUCIBILITY STATEMENT

The paper describes the benchmark construction, evaluation protocol, and T-MCTS training procedure. All code and datasets used in this work are available through the anonymous repository linked at the end of the abstract. These materials support independent reproduction of the benchmark and the reported evaluation results.

REFERENCES

Anthropic. Claude sonnet 4.6 system card. Technical report, Anthropic, February 2026. URL https://www-cdn.anthropic.com/bbd8ef16d70b7a1665f14f306ee88b53f686aa75.pdf. Accessed: 2026-03-13.

BIG bench authors. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=uyTL5Bvosj.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, 2022.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. Transformers as soft reasoners over language. *arXiv preprint arXiv:2002.05867*, 2020.

Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 337–340. Springer, 2008.

Bruce Frederiksen. Applying expert system technology to code reuse with pyke. *PyCon: Chicago*, 2008.

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

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Minlie Huang, Nan Duan, Weizhu Chen, et al. Tora: A tool-integrated reasoning agent for mathematical problem solving. In *International Conference on Learning Representations*, volume 2024, pp. 48362–48395, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Adrian Groza. Getting started with prover9 and mace4. In *Modelling Puzzles in First Order Logic*, pp. 1–9. Springer, 2021.

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

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 8154–8173, 2023.

Mehran Kazemi, Bahare Fatemi, Hritik Bansal, John Palowitch, Chrysovalantis Anastasiou, Sanket Vaibhav Mehta, Lalit K Jain, Virginia Aglietti, Disha Jindal, Yuanzhu Peter Chen, et al. Big-bench extra hard. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 26473–26501, 2025.

Shashank Kirtania, Priyanshu Gupta, and Arjun Radhakrishna. Logic-lm++: Multi-step refinement for symbolic formulations. In *Proceedings of the 2nd Workshop on Natural Language Reasoning and Structured Explanations (@ ACL 2024)*, pp. 56–63, 2024.

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

Bill Yuchen Lin, Ronan Le Bras, Kyle Richardson, Ashish Sabharwal, Radha Poovendran, Peter Clark, and Yejin Choi. Zebralogic: On the scaling limits of llms for logical reasoning. *arXiv preprint arXiv:2502.01100*, 2025a.

Haohan Lin, Zhiqing Sun, Sean Welleck, and Yiming Yang. Lean-star: Learning to interleave thinking and proving. In *International Conference on Learning Representations*, volume 2025, pp. 66041–66062, 2025b.

Heng Lin and Zhongwen Xu. Understanding tool-integrated reasoning. *arXiv preprint arXiv:2508.19201*, 2025.

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

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint arXiv:2007.08124*, 2020.

Nicholas Nethercote, Peter J Stuckey, Ralph Becket, Sebastian Brand, Gregory J Duck, and Guido Tack. Minizinc: Towards a standard cp modelling language. In *International conference on principles and practice of constraint programming*, pp. 529–543. Springer, 2007.

Theo Olausson, Alex Gu, Ben Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua Tenenbaum, and Roger Levy. 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*, pp. 5153–5176, 2023.

OpenAI. GPT-5.5 system card. System card, OpenAI, April 2026. URL `https://deploymentsafety.openai.com/gpt-5-5/gpt-5-5.pdf`. Reference model: GPT-5.5.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3806–3824, 2023.

Mihir Parmar, Nisarg Patel, Neeraj Varshney, Mutsumi Nakamura, Man Luo, Santosh Mashetty, Arindam Mitra, and Chitta Baral. Logicbench: Towards systematic evaluation of logical reasoning ability of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13679–13707, 2024.

Nisarg Patel, Mohith Kulkarni, Mihir Parmar, Aashna Budhiraja, Mutsumi Nakamura, Neeraj Varshney, and Chitta Baral. Multi-logieval: Towards evaluating multi-step logical reasoning ability of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 20856–20879, 2024.

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

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. Proofwriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the association for computational linguistics: ACL-IJCNLP 2021*, pp. 3621–3634, 2021.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Ziwei Chai, Y Charles, HS Che, Cheng Chen, et al. Kimi k2. 5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Ramya Keerthy Thatikonda, Wray Buntine, and Ehsan Shareghi. Logical reasoning with outcome reward models for test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 26113–26123, 2025.

Peng-Yuan Wang, Tian-Shuo Liu, Chenyang Wang, Ziniu Li, Yidi Wang, Shu Yan, Chengxing Jia, Xu-Hui Liu, Xinwei Chen, Jiacheng Xu, et al. A survey on large language models for mathematical reasoning. *ACM Computing Surveys*, 58(8):1–35, 2026.

Anjiang Wei, Yuheng Wu, Yingjia Wan, Tarun Suresh, Huanmi Tan, Zhanke Zhou, Sanmi Koyejo, Ke Wang, and Alex Aiken. Satbench: Benchmarking llms' logical reasoning via automated puzzle generation from sat formulas. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 33820–33837, 2025.

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

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Fangzhi Xu, Zhiyong Wu, Qiushi Sun, Siyu Ren, Fei Yuan, Shuai Yuan, Qika Lin, Yu Qiao, and Jun Liu. Symbol-llm: Towards foundational symbol-centric interface for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13091–13116, 2024.

Fangzhi Xu, Qika Lin, Jiawei Han, Tianzhe Zhao, Jun Liu, and Erik Cambria. Are large language models really good logical reasoners? a comprehensive evaluation and beyond. *IEEE Transactions on Knowledge and Data Engineering*, 37(4):1620–1634, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. Reclor: A reading comprehension dataset requiring logical reasoning. *arXiv preprint arXiv:2002.04326*, 2020.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. Rest-mcts*: Llm self-training via process reward guided tree search. *Advances in Neural Information Processing Systems*, 37:64735–64772, 2024.

Ming Zhang, Qiyuan Peng, Yinxi Wei, Yujiong Shen, Kexin Tan, Yuhui Wang, Zhenghao Xiang, Junjie Ye, Zhangyue Yin, Zhiheng Xi, et al. Llmeval-logic: A solver-verified chinese benchmark for logical reasoning of llms with adversarial hardening. *arXiv preprint arXiv:2605.19597*, 2026.

Wanjun Zhong, Siyuan Wang, Duyu Tang, Zenan Xu, Daya Guo, Jiahai Wang, Jian Yin, Ming Zhou, and Nan Duan. Ar-lsat: Investigating analytical reasoning of text. *arXiv preprint arXiv:2104.06598*, 2021.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*, 2023.

A APPENDIX

A.1 LOGICEVAL-𝕏 CONSTRUCTION

A.1.1 PROBLEM SOURCES

We assemble the seed problem pool from four types of sources: standardized reasoning tests, mathematical puzzle books, constraint-based puzzle collections, and discrete mathematics or formal logic textbooks. Historical LSAT analytical reasoning questions and GMAT practice materials provide rule-based questions in an assessment setting. Puzzle books and collections contribute problems involving deduction across multiple clues, while constraint libraries and textbooks provide structured assignment, logic, and combinatorial problems. Table 6 lists the sources used for collection.

In total, we collect 5,097 seed problems: 1,160 from standardized tests, 325 from mathematical puzzle books, 3,049 from constraint-based collections, and 563 from discrete mathematics and formal logic textbooks. These problems subsequently undergo quality filtering, difficulty screening, rewriting, and expert verification before inclusion in LOGICEVAL-𝕏.

Table 6: Sources used to collect seed problems for LOGICEVAL-𝕏.

Category	**Source**	**Relevant material**
Standardized tests	Official LSAT Practice Tests	Historical analytical reasoning questions
	GMAT Official Prep	Practice questions involving analytical reasoning
Mathematical puzzle books	*The Game of Logic*	Deductive logic problems
	Amusements in Mathematics	Mathematical and combinatorial puzzles
	The Canterbury Puzzles	Mathematical and logical puzzles
Constraint-based collections	CSPLib	Structured constraint satisfaction problems
	Grid Puzzles	Clue-based grid logic puzzles
	QED Night Puzzle Collection	Logic-grid, Zebra, and deduction puzzles
Textbooks	*Discrete Mathematics: An Open Introduction*	Logic, counting, and discrete-structure exercises
	Applied Discrete Structures	Discrete mathematics exercises
	Lectures in Discrete Mathematics	Logic and discrete mathematics problems
	Open Logic Project	Formal logic exercises
	forall x: UBC edition	Formalization and deductive logic exercises

A.1.2 QUALITY FILTERING

We use Qwen3-32B (Team, 2025) to review all 5,097 collected problems. The model checks whether the wording identifies relevant entities and attributes clearly, whether the stated conditions determine an answer, and whether existing options are ambiguous or admit more than one correct choice. It returns a keep-or-discard decision with a brief explanation in XML format. This screening removes 658 problems with unclear wording, missing conditions, ambiguous options, or non-unique answers, leaving 4,439 candidates. Figure 11 shows the prompt.

For retained problems that are not already single-answer multiple-choice questions, we use the answer supplied by the source as the correct option and construct distractors from model responses. We randomly sample from Llama-3.1-8B (Grattafiori et al., 2024); Qwen3-8B, Qwen3-14B, and Qwen3-32B (Team, 2025); DeepSeek-v4-Flash and DeepSeek-v4-Pro (Xu et al., 2026); Kimi-

Quality Filtering Prompt

```
You are a reviewer of logical reasoning problems. Check whether
the problem is clear, complete, and has one uniquely determined
answer. Solve it as needed. Do not rewrite it.

Discard the problem for any of these issues:
- Unclear wording: an entity, attribute, or rule has multiple
  reasonable interpretations.
- Missing conditions: information needed to solve the problem
  is absent.
- Ambiguous options: options overlap or express equivalent
  answers. Ignore this check if no options are provided.
- Non-unique answer: multiple answers satisfy all stated
  conditions.

Problem:
%{problem}

Return only XML:
<review>
  <decision>keep or discard</decision>
  <issue>none or the main issue above</issue>
  <reason>one brief sentence</reason>
</review>
```

Figure 11: Prompt used for initial quality filtering.

Algorithm 1 Conversion to single-answer multiple-choice format

Require: Problem p, source answer $a^\star$, target option count K, query budget B
$D \leftarrow \emptyset$
for $t = 1, \ldots, B$ **do**
  Sample a model m from the model pool
  $\hat{a} \leftarrow \textsc{Answer}(m, p)$
  if $\hat{a} \neq a^\star$ and $\hat{a} \notin D$ **then**
    $D \leftarrow D \cup \{\hat{a}\}$
  end if
  if $|D| = K - 1$ **then**
    return $\textsc{Shuffle}(\{a^\star\} \cup D)$
  end if
end for
return flag p for further review

K2.6 (Team et al., 2026); and GLM-5.2 (Zeng et al., 2026), asking each sampled model to solve the original problem without seeing the answer. Distinct incorrect responses become candidate distractors; repeated responses and responses matching the source answer are excluded. We shuffle the completed options and flag problems with insufficient distinct distractors for further review. The later expert-verification stage checks that each converted question has exactly one correct option. Algorithm 1 summarizes the procedure.

A.1.3 Human Difficulty Screening

We assess the 4,439 remaining candidates with 13 volunteers: five undergraduate students and eight graduate students from science and engineering disciplines. Each problem is randomly assigned to three different volunteers, who solve it independently. They see the problem statement and answer options without the source answer and submit one choice per problem.

Complexity Annotation Prompt

```
You are an annotator of logical reasoning problems. Read the
problem and count its complexity along three dimensions:

- Entity Count: distinct entities whose identities, relationships,
  or assignments matter to the solution.
- Attribute Slots: unknown attributes or assignments that must be
  determined.
- Stated Clues: explicit conditions given in the problem statement.

Count only information in the problem statement. Do not count
answer options or background details that impose no condition.

Problem:
%{problem}

Return only XML with integer values:
<annotation>
  <entity_count>integer</entity_count>
  <attribute_slots>integer</attribute_slots>
  <stated_clues>integer</stated_clues>
</annotation>
```

Figure 12: Prompt for the initial annotation of the three complexity dimensions.

For each problem p_i, we compute human accuracy from its three responses:

$$h_i = \frac{1}{3}\sum_{j=1}^{3}\mathbb{I}\big(\hat{a}_{ij} = a_i^\star\big),$$

where $\hat{a}_{ij}$ is volunteer j's answer and $a_i^\star$ is the source answer. We rank problems by h_i and use the lowest 20% as the selection threshold. Because three responses produce only four possible accuracy values, we retain all problems tied at the cutoff. This yields 1,023 candidates for complexity annotation.

A.1.4 Complexity Dimensions and LLM Annotation

We quantify the complexity of the 1,023 selected problems along three observable dimensions. An LLM reads each problem and assigns integer counts using the following definitions:

- *Entity Count* is the number of distinct entities whose identities, relationships, or assignments matter to the solution. A higher count means more entities must be distinguished and related.
- *Attribute Slots* is the number of unknown attributes or assignments that must be determined. A higher count means more values remain to be resolved.
- *Stated Clues* is the number of explicit conditions given in the problem statement. A higher count means more stated constraints must be satisfied.

The LLM excludes answer options and background details that impose no condition. Figure 12 shows the annotation prompt.

A.1.5 Expert Audit of Complexity Annotations

To assess the LLM-generated counts, we randomly audit 10% of the 1,023 selected candidates and 10% of each comparison benchmark in Table 1: ProofWriter (Tafjord et al., 2021), BIG-bench (bench authors, 2023), AR-LSAT (Zhong et al., 2021), FOLIO (Han et al., 2024), and ProntoQA (Saparov & He, 2022). Six mathematics and statistics experts, including faculty and graduate researchers, independently annotate the sampled problems. We assess inter-expert agreement before

Entity and Constraint Mapping Prompt

```
You are rewriting a logical reasoning problem into a different
theme. First create a one-to-one mapping for all entities and
attribute values needed to solve the problem. Then express every
stated clue and the correct answer under this mapping.

Preserve each clue's logical relation, order, negation, and
quantity. Do not add or remove conditions.

Original problem:
%{original_problem}

Source answer:
%{source_answer}

Return only well-formed XML:
<mapping>
  <new_theme>brief description</new_theme>
  <entity_pairs>
    <pair><original>...</original><new>...</new></pair>
  </entity_pairs>
  <attribute_pairs>
    <pair><original>...</original><new>...</new></pair>
  </attribute_pairs>
  <mapped_clues>
    <clue>...</clue>
  </mapped_clues>
  <mapped_answer>...</mapped_answer>
</mapping>
```

Figure 13: Prompt for mapping a problem to a new theme while preserving its logical structure.

resolving disagreements through discussion; ICC(2,6) is 0.80. We compare the LLM annotations with the adjudicated expert counts using exact-match rate and mean absolute error for each dimension.

A.1.6 Logic-preserving Rewriting

Entity and Constraint Mapping. For each selected problem, we first ask an LLM to construct a mapping from the original entities and attributes to counterparts in a different theme. The model applies this mapping to every stated clue and to the source answer. We require one-to-one correspondence between the original and new elements, while preserving the relations, ordering, negation, and quantities expressed by the clues. Figure 13 shows the mapping prompt.

Problem Rewriting. A second LLM receives the original problem and the generated mapping. It writes a fluent single-answer multiple-choice question in the new theme, incorporates the mapped clues, and rewrites and shuffles the answer options. The model returns both the rewritten question and the correct option in XML so that the answer remains identifiable after shuffling. The prompt is shown in Figure 14.

Iterative Logic Check. Finally, an LLM judge compares the original problem, mapping, and rewritten question. It checks whether each clue retains its meaning and whether the indicated answer remains uniquely correct. If it detects an error, it returns a corrected version, which is submitted to the same check again. A rewrite that passes this check proceeds to expert verification. Figure 15 gives the judging prompt.

Problem Rewriting Prompt

```
You are an editor of logical reasoning problems. Rewrite the
original problem using the supplied mapping. Make the new problem
clear and natural, include every mapped clue, and create distinct
answer options in a shuffled order.

Keep the mapped answer as the only correct option. Do not introduce
new facts or conditions.

Original problem:
%{original_problem}

Mapping:
%{mapping_xml}

Return only well-formed XML:
<rewrite>
  <problem>rewritten problem statement</problem>
  <options>
    <option label="A">...</option>
    <option label="B">...</option>
    <option label="C">...</option>
    <option label="D">...</option>
    ...
  </options>
  <correct_option>one option label</correct_option>
</rewrite>
```

Figure 14: Prompt for generating the rewritten question and its shuffled options.

A.1.7 Expert Verification

Six experts in mathematics and statistics, including faculty members and graduate researchers, verify the 1,023 rewritten problems. For each problem, they receive the original version, the rewritten version, the entity and attribute mapping, and the source answer. They independently check whether every stated clue retains its logical meaning, whether the rewritten options remain distinct, and whether the indicated answer is correct.

The experts also solve the rewritten problem using its stated clues and compare their answer with the indicated option. This check identifies rewrites that alter the solution, introduce ambiguity, or omit conditions needed to reach the answer. Fleiss' κ across expert judgments is 0.75. Disagreements are resolved through discussion using the original problem and mapping as references. Problems that do not pass verification are excluded. Of the 1,023 candidates, 944 enter the final LogicEval-$\mathbb{X}$ benchmark.

A.1.8 Illustrative Complex Reasoning Problem

Figure 16 illustrates the complexity of LogicEval-$\mathbb{X}$. Although the problem involves only three entities, their identities, family roles, and numerical attributes create nine interdependent Attribute Slots. The truth-telling rules and twelve statements further couple these assignments, requiring the solver to jointly reconcile logical and numerical constraints to identify the correct answer.

A.2 Evaluation Details

A.2.1 Evaluated Models

We evaluate 10 representative LLMs on LogicEval-$\mathbb{X}$, covering small-scale models, large-scale models, and closed-source frontier models. The evaluated models are Llama-3.1-8B (Grattafiori et al., 2024); Qwen3-8B, Qwen3-14B, and Qwen3-32B (Team, 2025); DeepSeek-v4-Flash and

Logic-preservation Check Prompt

```
You are checking whether a rewritten logical reasoning problem
preserves the original problem. Compare the clues, entity and
attribute mapping, answer options, and indicated correct answer.

If the rewrite changes a condition or the correct answer, correct
the rewritten problem and options. If it is logically consistent
with the original, return it unchanged.

Original problem:
%{original_problem}

Source answer:
%{source_answer}

Mapping:
%{mapping_xml}

Current rewrite:
%{candidate_xml}

Return only well-formed XML:
<judgment>
  <decision>pass or revise</decision>
  <issue>none or a brief description of the error</issue>
  <rewrite>
    <problem>accepted or corrected problem statement</problem>
    <options>
      <option label="A">...</option>
      <option label="B">...</option>
      <option label="C">...</option>
      <option label="D">...</option>
      ...
    </options>
    <correct_option>one option label</correct_option>
  </rewrite>
</judgment>
```

Figure 15: Prompt for checking and correcting the logical consistency of a rewrite.

DeepSeek-v4-Pro (Xu et al., 2026); Kimi-K2.6 (Team et al., 2026); GLM-5.2 (Zeng et al., 2026); Claude-Sonnet-4.6 (Anthropic, 2026); and GPT-5.5 (OpenAI, 2026).

For the API-based models, we evaluate DeepSeek-v4-Flash (2026-07-31), DeepSeek-v4-Pro (2026-08-13), Kimi-K2.6 (2026-04-20), GLM-5.2 (2026-06-17), Claude-Sonnet-4.6 (2026-02-17), and GPT-5.5 (2026-04-23). This selection spans different model scales and deployment settings.

A.2.2 Inference Settings and Deployment

All experiments are conducted under a unified inference protocol. Each problem in LogicEval-𝕏 is presented as a multiple-choice question, where models are instructed to first generate their reasoning process and then provide the final selected option. The sampling temperature is set to 1.0 and the maximum generation length is set to 4,096 tokens for all models.

Open-source models are deployed locally using eight NVIDIA A800 GPUs, while closed-source models are evaluated through their official APIs. All models use the same prompt template, decoding configuration, and answer extraction procedure to ensure fair comparisons.

For each problem, every model generates three independent responses. These responses are used to evaluate both answer accuracy and response consistency. All experiments on LogicEval-𝕏,

Example Logical Reasoning Problem

```
On Gua Island, the Honest Tribe always tells the truth, and the Lying
Tribe always lies. One member of each tribe marries the other. After
marriage, the former honest member tells one lie after every three
truths; the former liar tells one truth after every three lies. Their
son alternates between truth and lie, telling two of each in four
statements.

A, B, and C are the three family members. Their names are Jack,
Andrew, and Wallace, and they have different tribe numbers. Each makes
four statements:

A says:
  (1) Jack's tribe number is the largest.
  (2) I used to be from the Honest Tribe.
  (3) B is my wife.
  (4) My tribe number is 22 larger than B's.

B says:
  (1) A is my son.
  (2) My name is Jack.
  (3) C's tribe number is 54, 78, or 81.
  (4) C used to be from the Lying Tribe.

C says:
  (1) Andrew's tribe number is 10 larger than Wallace's.
  (2) A is my father.
  (3) A's tribe number is 66, 68, or 103.
  (4) B used to be from the Honest Tribe.

Which description is correct?

A) A: wife, Jack, 66; B: husband, Wallace, 44; C: son, Andrew, 54.
B) A: husband, Wallace, 44; B: wife, Jack, 66; C: son, Andrew, 54.
C) A: son, Andrew, 54; B: husband, Wallace, 44; C: wife, Jack, 66.
D) A: wife, Andrew, 54; B: husband, Jack, 66; C: son, Wallace, 44.

Correct answer: A
```

Figure 16: An example logical reasoning problem with identity and numerical constraints.

including baseline evaluation, symbolic tool integration, and $\mathbb{T}$-MCTS evaluation, follow the same inference protocol.

A.2.3 Evaluation Metrics

Accuracy. Accuracy measures the proportion of correctly answered problems:

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(\hat{y}_i = y_i), \tag{5}$$

where N denotes the number of evaluation problems, y_i represents the ground-truth option, and $\hat{y}_i$ denotes the predicted option.

Output Length. We measure the average number of generated tokens:

$$\text{Length} = \frac{1}{N}\sum_{i=1}^{N} T_i, \tag{6}$$

where T_i represents the number of generated tokens for problem i. For models whose internal reasoning traces are hidden by APIs, only visible final outputs are counted.

Self-consistency@3. To evaluate response stability, each model generates three independent responses for each problem. Self-consistency@3 is defined as:

$$\text{SC@3} = \frac{1}{N}\sum_{i=1}^{N} \frac{\max_y n_i(y)}{3}, \tag{7}$$

where $n_i(y)$ denotes the number of times option y is selected among the three responses for problem i.

Error Distribution. For error analysis, incorrect responses are categorized according to the taxonomy in Table 7. The proportion of each error type is computed as:

$$P(E_k) = \frac{N_k}{\sum_j N_j}, \tag{8}$$

where N_k denotes the number of incorrect responses assigned to error type k.

Tool Integration Gain. For symbolic tool integration experiments, we report the accuracy improvement:

$$\Delta Acc = Acc_{\text{tool}} - Acc_{\text{direct}}, \tag{9}$$

where Acc_{tool} and Acc_{direct} denote accuracy with and without symbolic tools, respectively.

A.2.4 Error Taxonomy

We define five mutually exclusive error types for analyzing incorrect model responses. Among them, Length Exhaustion, Reasoning Hallucination, and Constraint Collapse are treated as *Lost in Complexity*-related errors, since they reflect failures caused by prolonged reasoning, unstable constraint tracking, or invalid intermediate reasoning under high complexity.

A.2.5 Answer Extraction and Evaluation Protocol

For each model response, we extract the final selected option as the prediction result. Responses with malformed outputs, missing final options, or invalid answer formats are counted as incorrect.

For multiple independent responses generated by the same model, accuracy is computed over all responses, while Self-consistency@3 is computed based on agreement among the three generated answers. The same evaluation procedure is applied consistently across all experiments, including baseline comparisons, complexity analysis, symbolic tool integration, and $\mathbb{T}$-MCTS evaluation.

A.3 Training Details

This section describes the training details of Logic-$\mathbb{X}$-8B and Logic-$\mathbb{X}$-14B, including the training data, $\mathbb{T}$-MCTS trajectory construction, supervised fine-tuning, solver-feedback-driven reinforcement learning, and evaluation settings.

Table 7: Error taxonomy for incorrect responses on LOGICEVAL-𝕏.

Error Type	Definition	Lost in Complexity?
Format Error	The model fails to produce a valid final option or violates the required answer format, making the response unparsable. This includes missing answers, multiple conflicting final answers, or outputs outside the candidate options.	No
Length Exhaustion	The model continues reasoning until reaching the generation budget or context limit without completing a valid solution. The response often contains unfinished enumeration, repeated checking, or incomplete derivations.	Yes
Reasoning Hallucination	The model introduces unsupported conditions, entities, relations, or intermediate conclusions that are not stated or entailed by the problem. These hallucinated elements lead the reasoning process away from the valid solution space.	Yes
Constraint Collapse	The model fails to maintain previously established constraints or intermediate states. It may forget earlier conditions, violate known relations, or produce contradictions during multi-step reasoning.	Yes
Shortcut Reasoning	The model selects an answer using surface-level cues, option similarity, linguistic plausibility, or incomplete reasoning instead of solving the full logical problem. The response may be fluent but lacks sufficient constraint-based derivation.	No

A.3.1 BASE MODELS

We instantiate two symbolic tool-integrated reasoning models from instruction-tuned Qwen3 models. LOGIC-𝕏-8B is initialized from Qwen3-8B-Instruct, and LOGIC-𝕏-14B is initialized from Qwen3-14B-Instruct. Both models are optimized with full-parameter fine-tuning in the supervised fine-tuning stage and the reinforcement learning stage.

A.3.2 TRAINING DATA

The training pool contains 20,926 instances after preprocessing. Most training instances are self-collected logical reasoning problems. These include candidate problems collected during the construction of LOGICEVAL-𝕏 but not included in the final benchmark because they did not meet the high-difficulty threshold. The sources include Law School Admission Test (LSAT) and Graduate Management Admission Test (GMAT) analytical reasoning questions, mathematical logic puzzle books, constraint satisfaction puzzle collections, and discrete mathematics problem books. The composition of the training data is shown in Table 8.

Table 8: Data composition.

Source	Instances
Self-collected	16,864
AR-LSAT	1,585
FOLIO	1,001
ProntoQA	500
ProofWriter	488
Big-bench	488
Total	20,926

To increase source diversity, we also incorporate public logical reasoning datasets, including ProofWriter, LogicBench, AR-LSAT, FOLIO, and ProntoQA. Since these datasets differ substantially in scale, we use the available training instances from the smaller datasets and randomly sample from the larger ones. All instances are converted into a unified multiple-choice format before trajectory construction.

To avoid test leakage, we remove overlaps with LOGICEVAL-𝕏 using normalized string matching and similarity-based near-duplicate filtering. No separate validation split is used; the final checkpoints after the predefined training schedule are used for evaluation. LOGICEVAL-𝕏 is used only as the test set.

A.3.3 𝕋-MCTS TRAJECTORY CONSTRUCTION

For each training instance, we run 𝕋-MCTS three times, with a maximum of 16 rollouts per run and a maximum search depth of 6. The action space contains five tool-use operations: symbolic formulation, solver selection, solver invocation, feedback-based revision, and answer prediction. The symbolic solvers include PYKE, PROVER9, MINIZINC, and Z3. Each solver call is assigned a fixed timeout, and timeout cases are treated as non-executable trajectories. The main trajectory construction settings are listed in Table 9.

We retain trajectories whose length is within 4,096 tokens. Approximately 95% of generated trajectories fall within this length limit, and the average trajectory length is 1,857 tokens. A trajectory is considered solver-verified if the generated symbolic program can be executed by the selected solver and the answer is correct.

Table 9: $\mathbb{T}$-MCTS trajectory construction settings.

Item	Setting
Runs per case	3
Max rollouts per run	16
Max search depth	6
Max trajectory length	4,096 tokens
Average trajectory length	1,857 tokens
Length coverage	95% within 4,096 tokens
Action space	Formulate, Select, Call, Revise, Answer
Symbolic solvers	Pyke, Prover9, MiniZinc, Z3

For each case, we randomly retain one correct solver-verified trajectory for SFT. This strategy avoids reducing SFT to a single highest-return pattern and preserves diverse successful tool-use behaviors, such as solver reselection, symbolic revision, and feedback-based repair.

For each completed trajectory τ, we compute the solver-grounded return as:

$$R_T(\tau) = \mathrm{clip}\left(\alpha R_{\mathrm{ans}} + \beta R_{\mathrm{exec}} - \lambda_{\mathrm{switch}} n_{\mathrm{switch}} - \lambda_{\mathrm{repair}} n_{\mathrm{repair}}, 0, 1\right), \tag{10}$$

where R_{ans} indicates whether the final answer is correct, and R_{exec} indicates whether the generated symbolic program can be executed by the selected solver. n_{switch} and n_{repair} denote the number of solver reselection operations and feedback-based revision operations, respectively. We set $\alpha = 0.8$, $\beta = 0.2$, and $\lambda_{\mathrm{switch}} = \lambda_{\mathrm{repair}} = 0.05$. This reward prioritizes correct and executable symbolic solutions while discouraging unnecessary solver switching and repeated repair.

A.3.4 Trajectory-based Supervised Fine-Tuning

We first train the models on solver-verified $\mathbb{T}$-MCTS trajectories. This stage provides cold-start supervision for symbolic formulation, solver selection, solver invocation, feedback interpretation, and final answer prediction. Since one correct solver-verified trajectory is retained for each training instance, the SFT set contains 20,926 trajectories.

Both models are trained for 3 epochs with a global batch size of 128 and a maximum sequence length of 4,096 tokens. We use AdamW with a cosine learning rate schedule, a learning rate of 1×10^{-5}, a warmup ratio of 0.1, and a weight decay of 0.01. Training is conducted with bf16 precision and distributed full-parameter optimization. Under the batch size of 128, the number of SFT optimizer steps is 492.

Table 10: The GRPO training settings.

Item	Setting
Algorithm	GRPO
Group size G	8
Prompt batch size	32
Rollout batch	256 trajectories/update
Temperature	1.0
Top-p	1.0
Max generation length	4,096
Learning rate	1×10^{-6}
KL coefficient	0.001
Clip ratio	0.2
Online solver calls	Yes
Max RL epochs	2

A.3.5 Solver-feedback-driven Reinforcement Learning

After SFT, we construct model-specific RL training sets. Each SFT model performs five independent reasoning runs on every training instance. Cases answered correctly at least four times are removed as stable-correct cases, and cases never answered correctly are removed as stable-incorrect cases. The remaining cases, where the model answers correctly in one to three runs, are used for RL. This selection focuses reinforcement learning on instances where the model has partial reasoning capability but still lacks stable tool-use behavior. This procedure yields 5,123 RL cases for Logic-$\mathbb{X}$-8B and 3,821 RL cases for Logic-$\mathbb{X}$-14B. The GRPO training settings are listed in Table 10.

We optimize the models using GRPO with online solver feedback. For each problem, the model samples $G = 8$ tool-use trajectories. Each trajectory is evaluated with the solver-grounded return $R_T(\tau)$, and the group-normalized return is used as the advantage. Unlike critic-based reinforcement learning methods, we do not train an additional value model; the relative advantage is estimated from the sampled trajectory group. During RL, solver calls are executed online, allowing the model to use feedback from Pyke, Prover9, MiniZinc, and Z3 during trajectory generation and reward computation.

Table 11: Implementation and evaluation settings.

Implementation		Evaluation	
Item	**Setting**	**Item**	**Setting**
GPUs	8 NVIDIA A800 GPUs	Test set	LOGICEVAL-$\mathbb{X}$
SFT framework	LLaMA-Factory	Main decoding	Greedy decoding
RL framework	verl	Temperature	0
backend	vLLM	Top-p	1.0
Distributed training	DeepSpeed / FSDP	Max output length	4,096
Solver execution	Multi-process parallel calls	SC samples	3
Precision	bf16	SC temperature	0.7
Random seed	42	SC top-p	0.95

Table 12: Implementation and evaluation settings.

Implementation		Evaluation	
Item	**Setting**	**Item**	**Setting**
GPUs	8 NVIDIA A800 GPUs	Test set	LOGICEVAL-$\mathbb{X}$
SFT framework	LLaMA-Factory	Main decoding	Greedy decoding
RL framework	verl	Temperature	0
Infra framework	vLLM	Top-p	1.0
Distributed training	DeepSpeed / FSDP	Max output length	4,096
Solver execution	Multi-process parallel calls	SC samples	3
Precision	bf16	SC temperature	0.7
Random seed	42	SC top-p	0.95

We use a prompt batch size of 32, giving 256 sampled trajectories per update. The sampling temperature is set to 1.0, top-p is set to 1.0, and the maximum generation length is 4,096 tokens. The RL learning rate is 1×10^{-6}, with a KL coefficient of 0.001 and a clipping ratio of 0.2. Rewards are normalized within each sampled group. Training runs for at most 2 epochs and stops early if the moving-average reward and training accuracy show no improvement for 30 consecutive updates.

A.3.6 Implementation and Evaluation Settings

The implementation settings are summarized in the left part of Table 12. All training experiments are conducted on 8 NVIDIA A800 GPUs. We use LLaMA-Factory for supervised fine-tuning and verl for reinforcement learning, with vLLM used for trajectory generation. Full-parameter training is accelerated with DeepSpeed/FSDP, and solver calls are executed in parallel with a multi-process backend. Unless otherwise specified, training is performed with bf16 precision and a random seed of 42.

The evaluation settings are summarized in the right part of Table 12. We evaluate the final checkpoints on LOGICEVAL-$\mathbb{X}$ using the same prompt template and symbolic solver configuration as training. For the main results, we use greedy decoding with temperature 0, top-$p = 1.0$, and a maximum output length of 4,096 tokens. For Self-consistency@3, we sample three responses for each problem with temperature 0.7 and top-$p = 0.95$, and compute answer stability across the generated outputs.

A.4 Additional Experimental Results

Table 13 presents the full comparison between direct reasoning and symbolic tool-integrated reasoning across all evaluated LLMs. The results show that symbolic tool use is not uniformly beneficial for general-purpose LLMs. For smaller open-source models such as Llama-3.1-8B, Qwen3-8B, and Qwen3-14B, direct symbolic tool invocation substantially reduces accuracy and increases output length, suggesting that these models often fail to select appropriate solvers or construct executable symbolic programs. In contrast, stronger models such as DeepSeek-v4, Kimi-K2.6, GLM-5.2, Claude-Sonnet-4.6, and GPT-5.5 benefit from symbolic tools, but the gains remain moderate and are accompanied by longer outputs.

Table 13: Comparison between direct answer and symbolic tool-integrated reasoning across LLMs.

Model	Setting	Accuracy (%) ↑	Output Length ↓	Self-consistency@3 (%) ↑
LLAMA-3.1-8B	Direct	26.91	523	38.72
	w/ Symbolic Tools	0.74 (-26.17)	5420 (+4897)	-
QWEN3-8B	Direct	26.44	721	43.01
	w/ Symbolic Tools	10.34 (-16.10)	4394 (+3673)	38.72 (-4.29)
QWEN3-14B	Direct	24.58	428	46.53
	w/ Symbolic Tools	17.05 (-7.53)	1266 (+838)	41.24 (-5.29)
QWEN3-32B	Direct	28.57	522	55.74
	w/ Symbolic Tools	18.20 (-10.37)	1591 (+1069)	64.61 (+8.87)
DEEPSEEK-V4-FLASH	Direct	33.05	825	58.36
	w/ Symbolic Tools	42.69 (+9.64)	2154 (+1329)	67.84 (+9.48)
DEEPSEEK-V4-PRO	Direct	31.36	438	64.99
	w/ Symbolic Tools	41.76 (+10.40)	1936 (+1498)	70.86 (+5.87)
KIMI-K2.6	Direct	37.22	1207	62.71
	w/ Symbolic Tools	45.56 (+8.34)	2691 (+1484)	67.41 (+4.70)
GLM-5.2	Direct	37.34	339	60.84
	w/ Symbolic Tools	47.59 (+10.25)	1441 (+1102)	68.29 (+7.45)
CLAUDE-SONNET-4.6	Direct	46.61	467	67.98
	w/ Symbolic Tools	53.18 (+6.57)	2864 (+2397)	74.43 (+6.45)
GPT-5.5	Direct	52.65	-	71.02
	w/ Symbolic Tools	59.64 (+6.99)	-	77.51 (+6.49)
LOGIC-$\mathbb{X}$-8B (*Ours*)	Base	10.34	4394	38.72
	w/ $\mathbb{T}$-MCTS	53.60 (+43.26)	1670 (-2724)	78.43 (+39.71)
LOGIC-$\mathbb{X}$-14B (*Ours*)	Base	17.05	1266	41.24
	w/ $\mathbb{T}$-MCTS	**62.71 (+45.66)**	1665 (+399)	**84.82 (+43.58)**

The proposed Logic-$\mathbb{X}$ models exhibit a different pattern. With $\mathbb{T}$-MCTS, Logic-$\mathbb{X}$-8B improves from 10.34% to 53.60%, while Logic-$\mathbb{X}$-14B improves from 17.05% to 62.71%, achieving the best accuracy and self-consistency among all evaluated settings. These results suggest that effective symbolic tool-integrated reasoning requires more than simply exposing LLMs to external solvers. Instead, models need a structured mechanism for solver selection, executable feedback, and trajectory-level repair, which is directly supported by $\mathbb{T}$-MCTS.