LogicTrack: Auditing Reasoning Trajectories of Large Language Models with Formal Logic Solvers

Anonymous ACL submission

Abstract

Chain-of-Thought (CoT) reasoning has been shown to improve the performance of large language models (LLMs), yet existing optimization methods largely rely on outcome-based feedback, leaving the logical validity of intermediate reasoning steps largely unverified. To address the gap whereby LLMs arrive at correct final answers through logically flawed intermediate reasoning chains, we propose LogicTrack, a neuro-symbolic framework that audits reasoning trajectories by auto-formalizing each reasoning step into symbolic representations and verifying it with automated theorem provers. LogicTrack introduces Solver-Based Backtracking Reward (SBR), a step-wise scoring mechanism that quantifies logical soundness and guides backtracking tree search at inference time. We further extend LogicTrack to construct supervised fine-tuning (SFT) data with backtracking traces from its trajectories, enabling fine-tuned models to internalize step-wise auditing as an intrinsic capability. Extensive experiments across 8 reasoning benchmarks and 7 LLMs demonstrate that LogicTrack effectively improves both the verifiability of reasoning chains and final answer pass rate, thereby enhancing overall CoT quality and trustworthiness in high-stakes domains.

1 Introduction

Reasoning chains have greatly advanced the capacity of large language models (LLMs) to tackle complex tasks in coding (Gao et al., 2023), question answering (Lu et al., 2022), and logical reasoning (Xu et al., 2024). Recent models such as OpenAI-o1 (Jaech et al., 2024) and DeepSeek-R1 (Guo et al., 2025) further improve performance by encouraging the model to verbalize and extend its thinking trajectory before generating a final answer. However, producing a correct final answer is not sufficient on its own. If the intermediate reasoning steps contain logically invalid operations, users in high-stakes domains such as legal and scientific reasoning cannot trust its conclusions, even when the final answer happens to be correct (Turpin et al., 2023; Huang and Chang, 2023). This concern is especially important as most current methods use outcome-driven training, which does not guarantee that the reasoning path itself is sound.

This concern has motivated a growing body of research focused on the quality of intermediate reasoning steps. Existing work can be broadly divided into two categories. The first is training-free test-time self-refinement, where the LLM generates a response, receives feedback through a critic model or retrieval mechanism, and applies a search algorithm to regenerate the response based on that feedback (Yao et al., 2023a; Xie et al., 2023). The second is training-based process supervision, where process reward models (PRMs) are trained to evaluate the quality of individual reasoning steps, typically using human-annotated or model-generated step-level labels (Lightman et al., 2023a; Khalifa et al., 2023). However, methods in both categories ultimately rely on human feedback or LLM-based judges to produce feedback and reward signals. Such signals are non-deterministic for logical correctness, prone to inconsistency, and difficult to scale reliably across diverse domains.

More recent efforts have turned to neuro-symbolic approaches to make verification more reliable. These include using formal tools for symbolic checking of LLM outputs (Olausson et al., 2023; Pan et al., 2023), auto-formalization of natural language reasoning into solver-verifiable logic, and self-refinement guided by solver feedback (Quan et al., 2024; Singh et al., 2026). Step-wise formal verification for natural language reasoning remains underexplored. Recent LogicReward (Xu et al., 2025b) and VeriCoT (Feng et al., 2025) represent important initial steps in this direction. Concretely, both methods first generate a complete response and then apply solver-based step verifica-

tion retroactively, constructing SFT and DPO training data from the verified traces.

However, both methods perform step verification in a post hoc way. They require a complete reasoning chain and final answer to be generated first, and then audit each step retroactively. This paradigm has two limitations. First, once the full response has been produced, delayed verification raises safety concerns as flawed reasoning may have already been presented to or acted upon by users. Second, a logical error in an intermediate step can propagate through and corrupt subsequent reasoning steps, making after-the-fact correction both more difficult and less reliable.

These limitations naturally raise the question of whether we can audit LLMs reasoning trajectory during generation and intervene with timely backtracking to regenerate from the point of failure. To address this question, we introduce **LogicTrack**, a neuro-symbolic framework for generation-time auditing and improvement of CoT reasoning through a formally defined verification reward. Specifically, LogicTrack decomposes each NL reasoning step into premises, explanations, and conclusions, autoformalizes them into logic specifications, and uses the solver for verification. We define **Solver-based Backtracking Reward (SBR)**, a fine-grained step-level signal that combines premise fidelity with solver verification, derived from this verification process. LogicTrack enables a reward-guided backtracking search at inference time. When SBR of a step falls below a predefined threshold, the model triggers a backtrack and regenerates the reasoning from that step onward.

Beyond inference-time auditing, we further extend the application of LogicTrack to **self-backtracking supervised fine-tuning (SFT)**. Unlike prior self-correction methods that rely on synthetically injected errors to construct backtracking datasets (Yang et al., 2025a), LogicTrack pairs real model-generated flawed trajectories with solver-verified corrections. Fine-tuning on these trajectories with a dedicated backtrack token `<backtrack>` allows the model to internalize step-wise verification behavior, and we show that this benefit generalizes across multiple benchmarks and models. Our main contributions are as follows:

- We propose LogicTrack, a neuro-symbolic framework that audits NL reasoning trajectories via autoformalization and solver verification at inference time. LogicTrack includes a reward-guided backtracking mechanism that uses Solver-based Backtracking Reward to audit and guide model step-wise reasoning.
- We evaluate LogicTrack on eight reasoning benchmarks across seven models, showing consistent improvements in both final-answer correctness and intermediate-step validity. Further ablation studies and diverse variants of the search strategy confirm the generalizability of LogicTrack.
- We show that LogicTrack can be extended to construct high-quality backtracking SFT data. By introducing a dedicated backtrack token, the fine-tuned model internalizes step-wise verification behavior and achieves self-backtracking in reasoning.

2 Method: LogicTrack

Preliminaries. When models use natural language to infer logical reasoning tasks, their goal is to determine the logical relationship between a set of given premises $P = \{p_1, \ldots, p_n\}$ and a query Q. Each data sample is a triple $x = (P, Q, y)$, where $y \in \mathcal{Y}$ is the ground-truth label from a dataset-specific label set $\mathcal{Y}$. Although label names (e.g., *entailment/contradiction*, *yes/no/uncertain*) differ across datasets, each $\mathcal{Y}$ corresponds to one of three underlying logical relations: $P \models Q$, $P \models \neg Q$, or neither. Given a sample x, the model response is an ordered sequence of T reasoning steps $(s_1, s_2, \ldots, s_T)$ followed by a final prediction $\hat{y}$. To ensure the quality of both the final prediction and the intermediate reasoning, we propose LogicTrack to audit the logical validity of each intermediate step and trigger backtracking when necessary. Figure 1 illustrates the overall workflow of LogicTrack, which proceeds in three parts: decomposition and formal verification (§2.1), reward-guided search with backtracking (§2.2), and the extended application of LogicTrack as higher-quality training data for self-backtracking supervised fine-tuning (§2.3).

2.1 Decomposition and Formal Verification

Step Decomposition We specify in the LLM response format that each reasoning step needs to present new conclusion(s) derived from context C_i and explanation E_i, the details are explained below. Let $\mathcal{K}_i = \{q_1, \ldots, q_i\}$ denote the set of derived conclusions from the first i verified reasoning steps ($\mathcal{K}_0 = \emptyset$). Specifically, the step s_i can be decomposed into three components $s_i = (C_i, E_i, q_i)$,

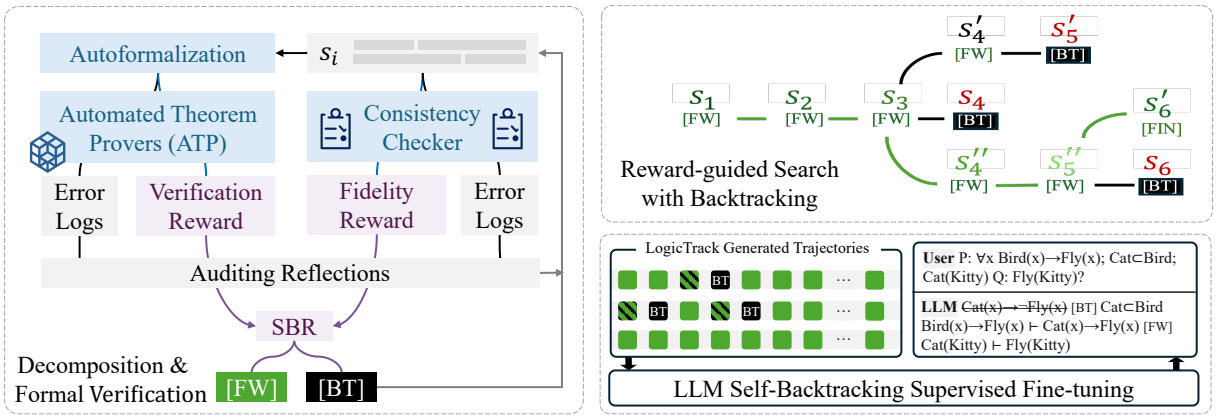

Figure 1: Overview of the LogicTrack workflow. Each NL reasoning step is decomposed, formalized into a solver-executable specification, verified, assigned an SBR score, and either forwarded [FW] to the next step or backtracked [BT] for regeneration. These generated trajectories can be used for backtracking fine-tuning.

defined as below. With both C_i and E_i stated explicitly, the conclusion q_i should follow by logical entailment alone. The verified q_i is then incorporated into $\mathcal{K}_i$, making it available as context for subsequent steps.

- **Context** C_i: the premises from P and prior verified conclusions relevant to this step, i.e., $C_i \subseteq P \cup \mathcal{K}_{i-1}$.
- **Explanation** E_i: implicit assumptions and background knowledge that the step relies on but does not state, e.g., "a father is a parent".
- **Conclusion** q_i: the new claim derived from C_i and E_i.

Autoformalization and Verification Unlike post-hoc verification methods that check the entire reasoning chain after the response has been fully generated, LogicTrack performs verification during reasoning in a step-wise autoregressive way. Starting from the $i-1$ verified steps $(s_1, \ldots, s_{i-1})$ and knowledge base $\mathcal{K}_{i-1}$, the reasoning LLM generates the next candidate step s_i, conditioned on $(P, Q, s_1, \ldots, s_{i-1})$. The candidate is then decomposed, formalized, and verified by the logic solver.

For each candidate step s_i, the formalization model receives the three components (C_i, E_i, q_i) and formalizes them into an executable logic specification F_i. The context and explanations are jointly formalized as a conjunction of solver assertions Φ_i, i.e., Φ_i encodes $C_i \cup E_i$. The conclusion q_i is encoded as the proof goal, yielding $F_i = (\Phi_i,\ q_i)$, so that verification reduces to checking $\Phi_i \models q_i$. In our implementation, the specification is produced in SMT-LIB format, so that every generated step can be checked immediately after it is proposed.

Given $F_i = (\Phi_i,\ q_i)$, LogicTrack verifies whether the candidate step is well-formed and logically justified. The solver first checks syntactic validity of the generated formalization; if the specification is malformed, the autoformalization module is re-invoked up to M times. Once a valid specification is obtained, the solver tests whether $\Phi_i \models q_i$ via proof by refutation, that is, it checks whether $\Phi_i \cup \{\neg q_i\}$ is unsatisfiable, and also records diagnostic signals such as contradiction or undecidability. These verification outcomes, together with the judge's assessment of explanation grounding, are stored as a step-level audit log and converted into the scalar reward used by the search controller.

2.2 Reward-Guided Backtracking Search

We consider two aspects of effective reasoning. The model must ground its reasoning in the given premises, and each inference must be logically sound. We design a reward function called the Solver-based Backtracking Reward (SBR) to quantify both aspects. LogicTrack assigns each step s_i a composite reward $\mathrm{SBR}(s_i) = \lambda \cdot R^{\text{fidelity}}(\Phi_i;\ P) + (1-\lambda) \cdot R^{\text{verify}}(F_i)$.

- **Fidelity reward** $R^{\text{fidelity}}(\Phi_i;\ P)$: Checks the consistency of Φ_i, including if each assertion is grounded in the original premises P (not sourced from the query Q), if introduced commonsense is acceptable, and if Φ_i is jointly satisfiable. We define $R^{\text{fidelity}}(\Phi_i;\ P) = 1$ if all checks pass, and 0 otherwise.
- **Verification reward** $R^{\text{verify}}(F_i)$: The verification score is set to 1 if the specification F_i is syntactically correct and the step is logically valid ($\Phi_i \models q_i$). If F_i is syntactically

invalid, autoformalization will be regenerated up to M times before the step is marked as unverifiable and $R^{\text{verify}}(F_i) = 0$. We also set $R^{\text{verify}}(F_i) = 0$ if the solver actively refutes the step ($\Phi_i \models \neg q_i$). For undecided steps ($\Phi_i \not\models q_i \wedge \Phi_i \not\models \neg q_i$), the verifier assigns partial credit $R^{\text{verify}}(F_i) = \alpha$, where $\alpha \in (0, 1)$ is a discounting constant, since these steps may simply exceed the solver's decision procedure without being logically wrong.

LogicTrack builds the verified reasoning chain via reward-guided backtracking search to decide whether to forward or backtrack on each candidate step. At each position t, the search maintains a verified prefix $(s_1, \ldots, s_{t-1})$ and evaluates a new candidate s_t against two quality criteria: the step-level reward $\text{SBR}(s_t)$ and running mean over all verified steps $\overline{\text{SBR}}(t) = \frac{1}{t}\sum_{i=1}^{t} \text{SBR}(s_i)$. The search starts from an empty prefix and proceeds one step at a time. A candidate s_t is forwarded if $\text{SBR}(s_t) \geq \theta_{\text{step}}$ and $\overline{\text{SBR}}(t) \geq \theta_{\text{avg}}$. In that case, s_t is appended to the verified prefix and $\mathcal{K}_{t-1}$ is updated to $\mathcal{K}_t$.

When triggering backtracking, LogicTrack collects the audit logs for s_t and re-prompts the model with the verified prefix $(s_1, \ldots, s_{t-1})$ along with concise failure feedback derived from error logs. The model then regenerates from position t onward. This backtracking process repeats until a viable solution is found. We provide implementation details and discussion of other backtracking strategies like Beam search and MCTS in Appendix A.1.

2.3 LogicTrack with Backtracking-SFT

We introduce LogicTrack-SFT, an extension of LogicTrack that converts its backtracking trajectories into supervised fine-tuning data, allowing the model to internalize step-level auditing as a intrinsic capability.

Recent work has shown that training on explicit error-correction trajectories can teach models to revise flawed reasoning. (An et al., 2023) construct mistake-correction pairs; (Ye et al., 2024) show that retry data with erroneous steps immediately followed by corrections improves reasoning; and Self-Backtracking (Yang et al., 2025b) further introduces a dedicated backtracking token to learn when and where to revise a partial trajectory. However, the supervision signals in these methods are largely synthesized from artificially injected or sampled errors in mathematical settings, where correctness is straightforward to verify. We build on the same

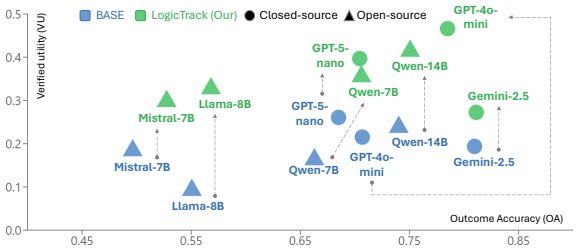

Figure 2: Overall OA and VU for all models averaged across eight benchmarks under BASE and LogicTrack settings. Here ● indicates closed source models, and ▲ indicates locally deployed models. Dashed arrows connect each model's BASE and LogicTrack results.

backtracking intuition but focus to a harder setting: using LogicTrack to construct verifier-grounded backtracking steps for NL logic reasoning.

Specifically, SFT training data is constructed from LogicTrack's backtracking trajectories. For each sample, we extract the verified correct path from the search tree alongside the backtracked steps that triggered backtracking events. A training trajectory is assembled by concatenating the correct prefix up to position $k-1$, the backtracked step s_k^-, a special backtrack token, and the verified correct continuation: $\{s_1^+ \cdots s_{k-1}^+ \;\; s_k^- \;\; \texttt{<BackTrack>} \;\; s_k^+ \cdots s_n^+\}$ Taking the workflow demo in Figure 1 as an example, one possible training trajectory with backtracking is $\{s_1, s_2, s_3, s_4, \texttt{<BackTrack>}, s_4'', s_5'', s_6'\}$. In the construction, we consider both single backtracking and multi-round backtracking. We additionally include clean correct trajectories and trajectories with multiple backtrack events, leading a set of ∼6,000 training samples. The backtrack token is added to the model vocabulary and trained end-to-end, teaching the model to recognize flawed steps and self-correct without external critics. The detailed implementations can be found in Appendix A.2.

3 Experiments and Results Discussion

This section introduces the experimental setup (§ 3.1), then presents LogicTrack's overall performance and step verifiability (§ 3.2), reports the ablation studies (§ 3.3), and the extended application of LogicTrack for SFT self-backtracking (§ 3.4).

3.1 Experimental Setup

Datasets and Baselines. We evaluate on a diverse collection of natural-language reasoning benchmarks including e-SNLI (Camburu et al., 2018), FOLIO (Han et al., 2024), LogiQA (Liu

Table 1: Benchmark-level Performance Comparison of VWA%, OA%, and UR% between BASE and LogicTrack (Ours) across Eight Reasoning Benchmarks and Seven Models.

Dataset	Metric	Gemini-2.5		GPT-4o-mini		GPT-5-nano		Llama-3.1-8B		Mistral-7B		Qwen2.5-7B		Qwen2.5-14B	
		BASE	Ours	BASE	Ours	BASE	Ours	BASE	Ours	BASE	Ours	BASE	Ours	BASE	Ours
FOLIO	VWA ⇑	21.17	56.37	46.35	59.91	46.35	51.36	27.76	39.40	32.87	43.82	31.90	48.97	45.14	57.53
	OA ⇑	79.31	77.59	68.97	74.14	71.70	73.28	48.28	45.69	48.28	54.31	59.48	65.52	75.00	75.00
	UR ⇓	75.63	27.79	34.74	19.76	38.75	31.90	41.85	17.45	33.75	26.72	47.65	28.41	42.09	28.00
ESNLI	VWA ⇑	31.09	39.08	46.38	87.21	30.10	44.32	11.89	19.07	16.12	22.74	35.25	50.52	39.50	53.89
	OA ⇑	66.67	66.67	86.05	100.00	65.12	63.57	22.48	21.71	31.01	31.01	60.47	64.34	68.22	65.89
	UR ⇓	52.84	42.96	45.87	12.79	53.36	29.46	38.45	11.18	35.84	21.64	40.71	23.58	40.61	15.67
LogiQA	VWA ⇑	24.91	20.36	34.33	44.98	24.30	40.23	25.97	44.72	28.11	36.27	24.81	40.38	29.17	46.70
	OA ⇑	62.50	65.91	61.93	54.55	50.86	55.11	46.02	50.00	44.32	46.02	51.70	51.70	55.11	62.50
	UR ⇓	63.07	68.56	46.26	16.05	54.82	31.70	42.25	12.47	39.62	23.48	50.17	25.20	50.43	27.22
Pron.QA	VWA ⇑	35.71	52.43	78.70	81.29	88.83	89.76	56.64	71.52	57.01	62.60	60.02	82.27	71.67	72.81
	OA ⇑	99.00	99.00	90.00	94.00	100.00	100.00	85.00	87.00	68.00	73.00	92.00	97.00	96.00	97.00
	UR ⇓	64.29	46.69	15.05	14.28	11.17	10.24	34.17	20.46	18.89	16.29	35.14	15.73	25.44	24.82
PW	VWA ⇑	32.45	53.61	40.21	63.82	75.41	78.02	36.65	42.91	19.33	29.02	41.95	53.58	48.35	49.61
	OA ⇑	88.44	86.43	54.27	79.90	100.00	99.50	59.80	57.29	27.14	39.20	63.32	66.83	80.40	78.39
	UR ⇓	65.13	36.97	37.45	23.82	24.59	21.47	40.77	29.10	38.31	26.83	38.19	22.35	43.59	38.15
QASC	VWA ⇑	20.78	55.88	54.43	71.35	30.82	43.04	47.89	64.02	25.67	40.64	0.00	0.00	59.09	80.59
	OA ⇑	86.30	91.78	93.15	90.41	61.64	63.01	84.93	75.34	53.42	60.27	87.67	87.67	91.78	94.52
	UR ⇓	76.14	44.12	42.60	23.29	42.58	26.71	44.65	16.30	56.06	33.56	100.00	100.00	37.95	16.67
SemEval	VWA ⇑	32.20	25.04	48.38	56.27	33.30	34.14	30.53	42.93	37.68	40.39	38.20	48.82	42.61	50.28
	OA ⇑	93.68	93.68	82.63	82.11	50.79	46.84	60.53	60.53	65.79	64.74	74.74	80.53	81.58	80.00
	UR ⇓	63.77	73.98	40.89	32.89	40.44	33.84	50.91	32.15	43.29	36.18	52.10	41.01	48.35	36.44
SARA	VWA ⇑	36.87	23.67	40.51	65.90	39.89	54.46	32.13	57.98	38.81	49.39	36.00	53.52	42.67	57.00
	OA ⇑	77.94	77.21	60.29	73.16	59.93	68.75	52.94	63.60	59.93	58.82	62.50	69.49	65.81	68.38
	UR ⇓	55.04	68.36	32.55	10.81	37.67	20.20	39.10	9.89	37.73	17.51	42.40	24.34	39.33	20.02

et al., 2020), ProntoQA (Saparov and He, 2023), ProofWriter (Tafjord et al., 2021), QASC (Khot et al., 2020), SARA (Holzenberger et al., 2020), and SemEval (SemEval-2026 Team, 2026). The evaluation covers seven large language models with varying architectures and scales, including three closed-source models (Gemini-2.5 (Comanici et al., 2025), GPT-4o (Hurst et al., 2024), and GPT-5 (Singh et al., 2025)) and four open-weight models (Llama (AI, 2024), Mistral (Jiang et al., 2023), Qwen2.5-7B, and Qwen2.5-14B (Yang et al., 2024)). The details are reported in Appendix B.1 and B.2.

Metrics. We evaluate models on four metrics, and these metrics consider outcome-based, process-based, and their combined performance. **Outcome Accuracy (OA)** is the fraction of LLM responses where the final answer is correct. **Verified Ratio (VR)** measures the average proportion of reasoning steps that pass solver checks. **Unverified Ratio (UR)** is the complementary fraction of unverified steps. **Verified Utility (VU)** refers to answer accuracy restricted to all steps fully-verified ones. **Verification-Weighted Accuracy (VWA)** combines both aspects by weighting each correct prediction by its per-problem verification ratio. More detailed metrics introductions are in Appendix B.3.

3.2 Better Outcome Correctness Performance and Step Verifiability

Figure 2 compares the average OA and VU of the base models and LogicTrack across all seven models, with detailed values reported in Table 3. Overall, LogicTrack consistently improves both metrics across all settings. The VU gains are especially pronounced: GPT-4o-mini rises from 21.51% to 46.61%, Llama3.1-8B from 9.24% to 32.75%, and Qwen2.5-7B from 16.33% to 35.54%. Among the closed-source models, GPT-4o-mini also achieves the strongest joint gain in OA, increasing from 70.68% to 78.49%.

Table 1 shows that the same pattern holds at the benchmark level. LogicTrack preserves or improves OA for most model and dataset pairs, with especially large gains on tasks that require multi-step deduction[1]. GPT-4o-mini on ProofWriter rises from 54.27% to 79.9%, Qwen2.5-7B on ProntoQA

[1]Qwen2.5-7B shows 0% VWA on QASC because it directly outputs answers with no reasoning steps, causing step-level evaluation to fail. This issue can be mitigated by our SFT model (Table 5).

Table 2: Ablation study of LogicTrack. (a) SBR reward component ablation. (b) Backtracking search strategy variants. The Δ report the absolute improvement (%) over BASE model.

(b) SBR Reward Component Ablation

Model	Method	VWA ⇑	ΔVWA ⇑	OA ⇑	ΔOA ⇑	UR ⇓
gpt-4o mini	Fidelity Only	45.80	5.3	70.96	10.7	37.19
	Solver Only	63.96	23.4	73.16	12.9	15.78
	SBR (Both)	65.90	25.4	73.16	12.9	10.81
Llama-3.1 8B	Fidelity Only	32.49	0.4	63.97	11.0	48.83
	Solver Only	57.55	25.4	62.50	9.6	7.89
	SBR (Both)	57.98	25.9	63.60	10.7	9.89

(a) Backtracking Search Strategy Variants

Model	Method	VWA ⇑	ΔVWA ⇑	OA ⇑	ΔOA ⇑	UR ⇓
gpt-4o-mini	Beam	54.87	14.4	69.12	8.8	23.79
	MCTS	53.06	12.6	72.06	11.8	26.84
	Main	65.90	25.4	73.16	12.9	10.81
Llama-3.1-8B	Beam	56.71	24.6	65.44	12.5	12.88
	MCTS	56.19	24.1	64.34	11.4	13.54
	Main	57.98	25.9	63.60	10.7	9.89

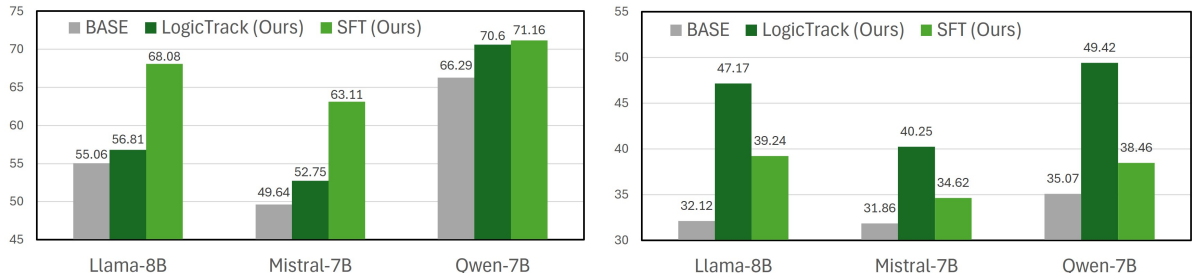

Figure 3: OA ⇑ and VR ⇑ under BASE, LogicTrack, and LogicTrack-SFT averaged over all datasets. LogicTrack-SFT refers to the model fine-tuned on LogicTrack-generated train data.

from 92% to 97%, and Mistral-7B on ProofWriter from 27.14% to 39.2%. Even when the base model is already strong, LogicTrack maintains OA while improving verifiability. GPT-5-nano on ProntoQA, for example, stays at high OA while UR drops from 11.17% to 10.24%. LogicTrack also reduces UR for most model and dataset pairs, confirming that backtracking replaces unverifiable steps with solver-validated alternatives. The largest reductions include Gemini-2.5 on FOLIO from 75.63% to 27.79%, Llama-3.1-8B on SARA from 39.1% to 9.89%, and Llama-3.1-8B on QASC from 44.65% to 16.3%. These reductions are accompanied by large VWA gains. GPT-4o-mini on ESNLI rises from 46.38% to 87.21%, and Gemini-2.5 on FOLIO from 21.17% to 56.37%. Overall, these results show that LogicTrack improves both final answers and the verifiability of the reasoning process.

3.3 Ablation Studies

We perform ablation studies on SARA to examine the effects of different reward designs and backtracking strategies on LogicTrack performance.

Effectiveness of SBR reward. Table 2b ablates the contribution of each SBR component: fidelity reward only R^{fidelity}, solver reward only R^{Verify}, and the full SBR reward. The fidelity reward R^{fidelity} yields stronger outcome-based metric gains, with a ΔOA of 10.7% on GPT-4o-mini and 11% on Llama-8B. Yet its effect on metrics that require step-level verification is far more modest. For instance, on the VWA metric, fidelity reward improves over the baseline by 5.4% on GPT-4o-mini and a mere 0.4% on Llama-8B. In contrast, the solver reward alone achieves VWA improvements exceeding 20% on both models. This suggests that grounding each step in the given premises helps the model avoid certain flawed steps and therefore contributes positively to final answer correctness. However, improving step-level verifiability still requires other reward signal.

This is validated by the results showing that reward R^{Verify} greatly improved the VWA performance. On GPT-4o-mini, the solver reward alone achieves 23.4% VWA and 12.9% OA gains over the baseline, while reducing the unverified ratio from 32.55% to 15.78%. On Llama-8B, it delivers an identical ΔVWA of 25.4% and brings the unverified step ratio down from 39.10% to 7.89%. These results align with our expectation that the solver reward directly incentivizes the model to produce structurally valid, executable reasoning steps, which are precisely what VWA and UR measure.

Combining both rewards yields the best VWA for both models with ΔVWA of 25.4% and 25.9% over the baseline, as well as the lowest UR on GPT-4o-mini at 10.81%. On Llama-8B, adding both ($R^{\text{fidelity}} + R^{\text{Verify}}$) further lifts OA from 62.50% to 63.60%, confirming that the two signals are complementary rather than redundant. R^{Verify} incentivizes structural correctness while R^{fidelity} enforces fac-

Example 1: Non-sequitur inference. *Premises:* Alex is a tumpus ⇒ jompus ⇒ dumpus ⇒ zumpus ⇒ yumpus; each yumpus is not cold. *Query:* Alex is cold. Answer label: FALSE

Before <backtrack>

Step 4: $\{C_4, E_4, q_4\}$, where q_4 : "Alex is not a wumpus (since zumpuses are dull and cannot be cold)."

× *Non-sequitur: "dull" is unrelated to "cold"; the chain must reach yumpus.*

After <backtrack>

Step 4: "Alex is a yumpus (every zumpus is a yumpus)."
Step 5: "Each yumpus is not cold ⇒ Alex is not cold."

✓ *Answer:* ***False***.

Example 2: Irrelevant derivation. *Premises:* Sally is a yumpus ⇒ zumpus ⇒ numpus; each numpus is not bitter. *Query:* Sally is not bitter. Answer label TRUE

Before <backtrack>

Step 1: $\{C_1, E_1, q_1\}$, where q_1 :"Sally is a yumpus, which means she is not opaque. But we need to determine if she is bitter or not."

× *Derives irrelevant property ("not opaque").*

After <backtrack>

Step 1: "Sally is a yumpus ⇒ zumpus ⇒ numpus."
Step 2: "Each numpus is not bitter ⇒ Sally is not bitter."

✓ *Answer:* ***True***.

Figure 4: Self-backtracking Examples on LogicTrack-SFT. Each pair contrasts the flawed step (left) with the corrected continuation (right).

tual grounding, and together they cover both dimensions of high-quality structural logical reasoning.

Backtracking Search Strategy Variants. Table 2a compares three search strategies, including default greedy backtracking, beam search, and monte carlo tree search. The defalt strategy achieves the highest VWA and lowest UR across both models. On GPT-4o-mini, it yields a ΔVWA of 25.4% and ΔOA of 12.9% over the baseline, with UR reduced to 10.81%. beam and MCTS search improves VWA by only 14.4% and 12.6% respectively and retain higher unverified ratios of 23.79% and 26.84%. On Llama-8B, all three strategies converge in VWA improvement at 25.9%, 24.6%, and 24.1%, yet the greedy strategy still achieves the lowest UR at 9.89%, compared to 12.88% and 13.54%.

We think this advantage is attributable to the greedy strategy's mechanism of immediate verification feedback. A failed step triggers instant delivery of the solver's audit log and regeneration from the exact failure position, whereas beam search and MCTS distribute their search budget across parallel candidates. Notably, MCTS achieves a higher ΔOA than beam search on GPT-4o-mini at 11.8% compared to 8.8%, suggesting that broader exploration benefits final answer correctness, though at the cost of weaker step-level verifiability.

3.4 Extended Results on LogicTrack-SFT

LogicTrack operates at inference time to improve step-level verification of model reasoning. We further extend it to the construction of fine-tuning datasets, exploring its potential for producing self-backtracking SFT data. We first report overall performance after SFT, then present case studies that compare reasoning steps before and after backtracking to show the effectiveness of the fine-tuning.

LogicTrack-SFT surpasses inference-time LogicTrack on answer correctness. As shown in Figure 3a, models fine-tuned on LogicTrack backtracking trajectories achieve the highest OA across all models. Compared with inference-time LogicTrack, SFT improves OA from 56.81% to 68.08% on Llama-8B, from 52.75% to 63.11% on Mistral-7B, and from 70.60% to 71.16% on Qwen-7B. This suggests that training on solver-verified trajectories with explicit backtrack tokens enables the model to internalize reasoning patterns that generalize beyond what inference-time search alone achieves.

SFT partially transfers step-level verification capability, though inference-time LogicTrack retains the advantage. Figure 3b shows that SFT also improves step-level quality: VR increases over the base model from 32.12% to 39.24% on Llama-8B, from 31.86% to 34.62% on Mistral-7B, and from 35.07% to 38.46% on Qwen-7B, showing that LogicTrack trajectories can transfer partial verification capability into the model's intrinsic reasoning. However, inference-time LogicTrack still achieves the highest VR, reaching 47.17%, 40.25%, and 49.42% on the three models respectively. This gap indicates that keeping the solver in the loop at inference time provides stronger step-level guarantees than the model's internalized self-correction alone, and suggests that distilling verification more effec-

tively from the solver into model parameters is a promising direction for future work.

Case Study of Self-Backtracking Corrections. Figure 4 presents two representative examples of self-backtracking from LogicTrack-SFT (Qwen2.5-7B). The trajectories are simplified here and the detailed example can be found in Appendix C.1. In Example 1, the model makes a non-sequitur by deriving an unrelated property instead of continuing the type chain. After backtracking, it completes the correct deduction. Example 2 shows an irrelevant derivation in which the model explores an off-topic property. The model triggers backtracking in these cases and after backtracking, the model follows the targeted reaosning path. These cases highlight LogicTrack-SFT's internalized backtracking ability and show how it recovers from non-sequitur inferences and unproductive reasoning paths in a way that is consistent with the solver-guided corrections in LogicTrack training trajectories.

4 Related Work

LLMs Reasoning Capability Enhancement. Many methods have been proposed to improve LLMs reasoning patterns, broadly falling into training-based and training-free methods. Training-free approaches improve reasoning quality during inference without updating parameters, including prompting strategies (Wang et al., 2022; Wei et al., 2022; Lightman et al., 2023b), search-based methods (Yang et al., 2025b; Yao et al., 2023b), and external knowledge augmentation (Peng et al., 2023). Training-based methods design rewards to identify high-quality reasoning data for fine-tuning (Gulcehre et al., 2023; Zelikman et al., 2022), or incorporate these rewards into the training objective (Luong et al., 2024). Reward designs include outcome-based and process-level critics that attend to intermediate steps (Yue et al., 2026; Calanzone et al., 2024; Lightman et al., 2023a; Wang et al., 2024). However, the criteria used to assess reasoning quality generally lack formal grounding. Outcome-based critics evaluate only final-answer correctness, while process-level critics rely on human annotations or LLM-generated judgments, limiting verifiability and allowing subtle logical errors to go undetected. To address this, we define intermediate-step reasoning quality based on whether each step is logically supported by deterministic step-level feedback from solver verification.

LLMs Reasoning Chain Verification. Recent work increasingly combines LLMs with verification mechanisms to improve soundness of model-generated reasoning (Cheng et al., 2025; Liu et al., 2025), generally falling into two paradigms: formalizing LLM responses and verifying them with external solvers (Olausson et al., 2023; Pan et al., 2023), or treating LLMs themselves as symbolic provers (Xu et al., 2025a, 2024). This work focuses on external solver-based approaches, as they provide a more explicit and controllable verification signal than self-verification. Within this paradigm, (Zhang et al., 2025; Ranaldi et al., 2025) discuss efficient auto-formalization; (Xu et al., 2026) study adaptive solver selection; (Quan et al., 2024, 2025) use solver feedback for iterative refinement. These works mainly target reasoning tasks close to formal representations, or highly structured domains such as mathematics and programming (Ren et al., 2025), where complete responses can be directly passed to a solver for verification. Step-level verification for general natural language reasoning remains under-explored, as it requires independently formalizing and checking each intermediate chain rather than one-off final answer verification. In contrast, LogicTrack verifies each natural language step during generation and backtracks at the first failed step, preventing local errors from propagating through the full reasoning chain.

5 Conclusion

We presented LogicTrack, a neuro-symbolic framework for generation-time auditing of LLM reasoning trajectories. LogicTrack decomposes each step into explicit context, explanations, and conclusions, autoformalizes them into solver-executable logic specifications, and uses the Solver-based Backtracking Reward (SBR) to combine premise fidelity with solver feedback for online backtracking and regeneration. Across eight reasoning benchmarks and seven LLMs, LogicTrack substantially improves step-level verifiability and generally preserves or improves final-answer accuracy, with ablations showing that solver verification drives most of the gains. LogicTrack backtracking traces also serve as useful supervision for fine-tuning open-source models, improving both final answer accuracy and verification performance over the base models. Future work on verification reward variants, backtracking strategies, and fine-tuning approaches that enable LLMs to internalize verification capabilities will be valuable.

Limitations

While LogicTrack presents effective improvements in step-level verifiability and final-answer correctness, several limitations should be noted. Our reward design uses a fixed weighting coefficient λ between fidelity and verification reward, which may not generalize optimally across reasoning domains. Investigating dynamic reward weighting in future work would extend the applicability of external solver verification to a broader range of domains. Our method relies on external solvers, and the additional verification at inference time introduces extra latency, limiting its applicability in real-time scenarios. Although our extended experiments show that verification capability can be partially internalized into LLMs via SFT, the fine-tuned models still underperform the inference-time method that leverages external solvers. Future exploration of more effective approaches to internalize formal verification capability into model parameters, closing the trade-off with external solvers while preserving inference efficiency would be valuable.

References

Meta AI. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. 2023. Learning from mistakes makes llm better reasoner. *arXiv preprint arXiv:2310.20689*.

Diego Calanzone, Stefano Teso, and Antonio Vergari. 2024. Logically consistent language models via neuro-symbolic integration. *arXiv preprint arXiv:2409.13724*.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: Natural language inference with natural language explanations. *Advances in Neural Information Processing Systems*, 31.

Fengxiang Cheng, Haoxuan Li, Fenrong Liu, Robert van Rooij, Kun Zhang, and Zhouchen Lin. 2025. Empowering llms with logical reasoning: A comprehensive survey. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 10400–10408.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Yu Feng, Nathaniel Weir, Kaj Bostrom, Sam Bayless, Darion Cassel, Sapana Chaudhary, Benjamin Kiesl-Reiter, and Huzefa Rangwala. 2025. Vericot: Neuro-symbolic chain-of-thought validation via logical consistency checks. *arXiv preprint arXiv:2511.04662*.

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

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, and 1 others. 2023. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*.

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

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, and 1 others. 2024. Folio: Natural language reasoning with first-order logic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22017–22031.

Nils Holzenberger, Andrew Blair-Stanek, and Benjamin Van Durme. 2020. A dataset for statutory reasoning in tax law entailment and question answering. *arXiv preprint arXiv:2005.05257*.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the association for computational linguistics: ACL 2023*, pages 1049–1065.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Muhammad Khalifa, Lajanugen Logeswaran, Moontae Lee, Honglak Lee, and Lu Wang. 2023. Grace: Discriminator-guided chain-of-thought reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15299–15328.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. 2020. Qasc: A dataset for question answering via sentence composition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8082–8090.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023a. Let's verify step by step. In *The twelfth international conference on learning representations*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023b. Let's verify step by step. In *The twelfth international conference on learning representations*.

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

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

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in neural information processing systems*, 35:2507–2521.

Trung Quoc Luong, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. 2024. Reft: Reasoning with reinforced fine-tuning. *arXiv preprint arXiv:2401.08967*.

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

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

Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, and 1 others. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint arXiv:2302.12813*.

Xin Quan, Marco Valentino, Louise A. Dennis, and Andre Freitas. 2024. Verification and refinement of natural language explanations through LLM-symbolic theorem proving. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2933–2958.

Xin Quan, Marco Valentino, Louise A. Dennis, and Andre Freitas. 2025. Faithful and robust LLM-driven theorem proving for NLI explanations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 17734–17755.

Leonardo Ranaldi, Marco Valentino, and Andre Freitas. 2025. Improving chain-of-thought reasoning via quasi-symbolic abstractions. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17222–17240.

ZZ Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, and 1 others. 2025. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*.

Abulhair Saparov and He He. 2023. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *Proceedings of the International Conference on Learning Representations*.

SemEval-2026 Team. 2026. Semeval-2026 task 11: Disentangling content and formal reasoning in large language models. `https://sites.google.com/view/semeval-2026-task-11`. Accessed: 2026-03-28.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, and 1 others. 2025. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*.

Vikash Singh, Darion Cassel, Nathaniel Weir, Nick Feng, and Sam Bayless. 2026. Verge: Formal refinement and guidance engine for verifiable llm reasoning. *arXiv preprint arXiv:2601.20055*.

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

Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. 2023. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36:74952–74965.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

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

Jundong Xu, Hao Fei, Meng Luo, Qian Liu, Liangming Pan, William Yang Wang, Preslav Nakov, Mong-Li Lee, and Wynne Hsu. 2025a. Aristotle: Mastering logical reasoning with a logic-complete decompose-search-resolve framework. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3052–3075.

Jundong Xu, Hao Fei, Liangming Pan, Qian Liu, Mong-Li Lee, and Wynne Hsu. 2024. Faithful logical reasoning via symbolic chain-of-thought. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 13326–13365.

Jundong Xu, Hao Fei, Huichi Zhou, Xin Quan, Qijun Huang, Shengqiong Wu, William Yang Wang, Mong-Li Lee, and Wynne Hsu. 2025b. Logicreward: Incentivizing llm reasoning via step-wise logical supervision. *arXiv preprint arXiv:2512.18196*.

Lei Xu, Pierre Beckmann, Marco Valentino, and André Freitas. 2026. Adaptive llm-symbolic reasoning via dynamic logical solver composition. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1187–1208.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, and 1 others. 2024. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Xiao-Wen Yang, Xuan-Yi Zhu, Wen-Da Wei, Ding-Chu Zhang, Jie-Jing Shao, Zhi Zhou, Lan-Zhe Guo, and Yu-Feng Li. 2025a. Step back to leap forward: Self-backtracking for boosting reasoning of language models. *arXiv preprint arXiv:2502.04404*.

Xiao-Wen Yang, Xuan-Yi Zhu, Wen-Da Wei, Ding-Chu Zhang, Jie-Jing Shao, Zhi Zhou, Lan-Zhe Guo, and Yu-Feng Li. 2025b. Step back to leap forward: Self-backtracking for boosting reasoning of language models. *arXiv preprint arXiv:2502.04404*.

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

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

Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. 2024. Physics of language models: Part 2.2, how to learn from mistakes on grade-school math problems. *arXiv preprint arXiv:2408.16293*.

Chuhuai Yue, Chengqi Dong, Yinan Gao, Hang He, Jiajun Chai, Wei Lin, and Guojun Yin. 2026. Promoting efficient reasoning with verifiable stepwise reward. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 34530–34538.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488.

Lan Zhang, Marco Valentino, and André Freitas. 2025. Masa: Llm-driven multi-agent systems for autoformalization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 615–624.

A Methods Implementation

A.1 Backtracking Search Strategies

This section provides additional implementation details for the search-strategy ablation in Table 2(a). We compare three variants: the default greedy backtracking search used in the main experiments, beam search, and Monte Carlo Tree Search (MCTS).

Default Search The default search is the greedy backtracking strategy used throughout the main experiments and the implementation is described in Section 2.2. Its step-level acceptance threshold is set to $\theta_{\text{step}} = 0.8$, the running-average threshold is set to $\theta_{\text{avg}} = 0.5$. The maximum number of regeneration attempts per step position is $k = 2$. If all k attempts fail, the candidate with the highest SBR is force-forwarded to guarantee forward progress.

Beam Search Beam search maintains a frontier of B partial reasoning paths and expanding all of them in parallel at each layer. At each layer, every frontier path contributes up to $k = 2$ continuation branches: if the path carries a pending tail from a prior generation, its leading provisional step is consumed as one free branch; the remaining slots are filled by fresh LLM completions conditioned on the current verified prefix. All verification tasks are dispatched concurrently. After verification, nodes with $\mathrm{SBR}(s_t) < 0.75$ are pruned; the top-B survivors ranked by path-average SBR advance to the next layer. If all candidates are pruned, the best low-reward node is reactivated to guarantee forward progress. The search terminates as soon as the top frontier node yields a complete reasoning chain; otherwise the best node found is returned.

Monte Carlo Tree Search We follow the standard MCTS framework, repeating four stages each iteration. During selection, the algorithm traverses from the root by choosing at each node the child with the highest UCB score, which balances the mean backpropagated SBR against an exploration bonus, until it reaches a node with fewer than k children. During expansion, up to k fresh LLM completions are generated from the selected node. The first step s_t of each completion is verified and attached as a new child. A child is marked terminal if $\mathrm{SBR}(s_t) = 0$, if the reasoning chain is complete, or if a depth limit is reached. During rollout, each non-terminal child whose $\mathrm{SBR}(s_t) \geq \theta$ is extended by consuming the remaining provisional steps from the same generation, verifying each in turn without additional LLM calls, until the chain terminates or a step falls below θ. During backpropagation, the path-average SBR of the reached leaf is propagated back to the root, updating each ancestor's statistics. After a fixed number of iterations the best complete leaf is returned, or, if none exists, the best leaf overall.

A.2 Backtracking SFT

We build the SFT data from backtracking traces produced by LogicTrack. For each training example, we keep only cases whose final answer is correct and whose reasoning steps pass both syntactic and semantic verification. From each trace, we extract the verified path $\mathbf{s}^+ = (s_1^+, \ldots, s_n^+)$ from `final_path` and the rejected steps $\{s_k^-\}$ collected at backtracking points. A rejected step at position k is kept only when its prefix up to step $k-1$ matches the verified path. This keeps the error local and avoids trajectories that have already diverged earlier.

We then construct three kinds of training trajectories. **(1) Single backtracking correction.** One wrong step is followed by a dedicated backtracking token and then the verified continuation,

$$s_1^+ \cdots s_{k-1}^+ \;\; s_k^- \;\; \texttt{<backtrack>} \;\; s_k^+ \cdots s_n^+ \;\; \boxed{\hat{y}}.$$

(2) Multiple backtracking corrections. These trajectories contain two or more corrections at different step positions. We prefer cases whose wrong continuation ends with an incorrect final answer. **(3) Clean correct trajectories.** These contain all verified steps and provide positive examples.

During fine tuning, we add `<backtrack>` as the new vocabulary item. The model learns to emit this token when a step is wrong and then continue with a corrected path, without relying on an external verifier at inference time. The final training set (`sft_cmp_bt_all`) contains about 6,000 samples across the three trajectory types. The control set (`sft_cmp_nobt_all`) contains about 3,800 samples without backtracking.

We fine tune three large language models on the constructed SFT dataset, Qwen2.5-7B-Instruct, Mistral-7B-Instruct-v0.3, and Llama-3.1-8B-Instruct. For all runs, we add `<backtrack>` to the tokenizer vocabulary. We use LoRA supervised fine tuning for 2 epochs with rank 32 and scaling factor 64. Training uses bfloat16 and AdamW (`adamw_torch`) with a learning rate of 3×10^{-5}, a linear scheduler, warmup ratio 0.05, and gradient clipping with maximum norm 0.3. The per device batch size is 2 and the effective batch size is 16. We also enable gradient checkpointing and save checkpoints every 50 steps.

B Experimental Settings

B.1 Datasets

We evaluate on eight reasoning benchmarks. For e-SNLI, FOLIO, LogiQA, ProntoQA, ProofWriter, and QASC, we adopt the processed releases used in LogicReward (Xu et al., 2025b). We then resplit their released 'data.jsonl' into training and test subsets with the ratio of 4 : 1. For SemEval (SemEval-2026 Team, 2026), Task 2 is used for testing and English portion of Task-4 is used for training. For SARA (Holzenberger et al., 2020), we follow the instruction from (Feng et al., 2025) and use the

instances from SARA Entailment in testing stage only.

We briefly summarize the benchmarks below.

- **e-SNLI** (Camburu et al., 2018) extends SNLI with human-annotated natural language explanations for premise–hypothesis pairs.
- **FOLIO** (Han et al., 2024) is a human-annotated benchmark for natural language reasoning with accompanying first-order logic annotations that are automatically verified by an inference engine.
- **LogiQA** (Liu et al., 2020) is originally a multiple-choice reading comprehension benchmark for logical reasoning over short passages. We use preprocessed version from (Xu et al., 2025b).
- **ProntoQA** (Saparov and He, 2023) is a synthetic question answering benchmark in which each example is generated from a synthetic world model represented in first-order logic, providing a controlled setting for analyzing multi-step deduction.
- **ProofWriter** (Tafjord et al., 2021) is a synthetic benchmark over natural-language theories of facts and rules, with associated questions and proofs at different reasoning depths.
- **QASC** (Khot et al., 2020) is an 8-way multiple-choice science question answering benchmark for question answering via sentence composition.
- **SARA** (Holzenberger et al., 2020) is a benchmark for statutory reasoning in tax law, with entailment and question answering tasks.
- **SemEval** (SemEval-2026 Team, 2026) refers to SemEval-2026 Task 11 on disentangling content from formal reasoning in multilingual syllogistic arguments. We use Task 2 for testing and the English portion of Task 4 for training, and in both cases use the binary validity labels.

B.2 Models

Our experiments cover seven models with diverse architectures and scales. For all models, the temperature is set to 0.7 and the maximum generation length is set to 10240 tokens. For closed-source models, experiments are performed via corresponding OpenAI and Gemini APIs. For open-weight models, all testing and fine-tuning experiments are run on NVIDIA A100 (40 GB / 80 GB) or equivalent GPUs.

Closed source models.

- **GPT-4o-mini** (Hurst et al., 2024) is a member of the GPT-4o family that offers strong reasoning performance at low inference cost.
- **GPT-5-nano** (Singh et al., 2025) is the smallest model in the GPT-5 family and is designed for fast and lightweight inference.
- **Gemini-2.5-Flash-Lite** (Comanici et al., 2025) is a lightweight model from the Gemini 2.5 family that balances efficiency with strong reasoning ability.

Open weight models.

- **Llama-3.1-8B-Instruct** (AI, 2024) is an instruction tuned model with 8B parameters from Meta's Llama 3.1 family.
- **Mistral-7B-Instruct-v0.3** (Jiang et al., 2023) is an instruction tuned model with 7B parameters from Mistral AI with strong performance relative to its size.
- **Qwen2.5-7B-Instruct** (Yang et al., 2024) is an instruction tuned model with 7B parameters from the Qwen2.5 family.
- **Qwen2.5-14B-Instruct** (Yang et al., 2024) is an instruction tuned model with 14B parameters from the same Qwen2.5 family with stronger reasoning capacity at higher computing cost.

B.3 Metric Definitions

For each problem x, let T_x be the number of reasoning steps and K_x the number of verified steps, giving the per-problem verification ratio $\rho_x = K_x/T_x$.

Outcome Accuracy (OA) is the fraction of examples with correct final answers. Verified Ratio (VR) is the dataset-average verification ratio $\frac{1}{N}\sum_x \rho_x$, Unverified Ratio (UR) is the mean fraction of unverified steps $\frac{1}{N}\sum_x (1-\rho_x)$.

Verified Utility (VU) is the answer accuracy restricted to problems whose reasoning chains are fully verified: $\mathrm{VU} = \frac{\sum_x \mathbf{1}[\rho_x=1]\cdot\mathbf{1}[\hat{y}_x=y_x]}{\sum_x \mathbf{1}[\rho_x=1]}$.

Verification-Weighted Accuracy (VWA) is defined as $\frac{1}{N}\sum_x \rho_x \cdot \mathbf{1}[\hat{y}_x = y_x]$, it weights each correct prediction by its per-problem verification ratio.

C Results

Table 3 reports the average OA and VU of the base models and LogicTrack across all seven models. Tables 4, 5, and 6 report the performance of the three LogicTrack-SFT models on each dataset.

C.1 Backtracking Case Study in LogicTrack-SFT

We expand Example 2 in Figure 4 with the full generated traces. We show the original flawed step that triggers <backtrack> and the regenerated continuation.

Original Given premises. P=' Every yumpus is not opaque; Yumpuses are zumpuses; Zumpuses are nervous; Every zumpus is a tumpus; Each tumpus is not large; Tumpuses are impuses; Impuses are temperate; Impuses are numpuses; Every wumpus is bitter; Each numpus is not bitter; Numpuses are vumpuses; Every vumpus is not kind; Vumpuses are rompuses; Rompuses are brown; Rompuses are jompuses; Every jompus is wooden; Every jompus is a dumpus; Sally is a yumpus.' **Query:** Sally is not bitter. **Answer label:** TRUE

```
Step 1:
##Premises:## Sally is a yumpus; Every yumpus
    is not opaque; Yumpuses are zumpuses;
    Zumpuses are nervous; Every zumpus is a
    tumpus; Each tumpus is not large; Tumpuses
    are impuses; Impuses are temperate; Impuses
    are numpuses; Every wumpus is bitter; Each
    numpus is not bitter; Numpuses are
    vumpuses; Every vumpus is not kind;
    Vumpuses are rompuses; Rompuses are brown;
    Rompuses are jompuses; Every jompus is
    wooden; Every jompus is a dumpus.
##Derived Fact:## Sally is a yumpus, which
    means she is not opaque. However, we need
    to determine if she is bitter or not.
<backtrack>
Step 1:
##Premises:## Sally is a yumpus; Every yumpus
    is not opaque; Yumpuses are zumpuses;
    Zumpuses are nervous; Every zumpus is a
    tumpus; Each tumpus is not large; Tumpuses
    are impuses; Impuses are temperate; Impuses
    are numpuses; Every wumpus is bitter; Each
    numpus is not bitter; Numpuses are
    vumpuses; Every vumpus is not kind;
    Vumpuses are rompuses; Rompuses are brown;
    Rompuses are jompuses; Every jompus is
    wooden; Every jompus is a dumpus.
##Derived Fact:## Sally is a yumpus. Since
    yumpuses are zumpuses and zumpuses are
    numpuses, we can conclude that Sally is
    also a numpus.
Step 2:
##Premises:## Sally is a numpus; Each numpus is
    not bitter.
##Derived Fact:## Sally is not bitter.
Step 3:
##Premises:## The hypothesis states: Sally is
    not bitter; We have established that Sally
    is not bitter.
##Derived Fact:## The hypothesis matches the
    derived fact above. Therefore, the claim is
    true. The final answer is \boxed{True}.
```

Table 3: Overall performance for all models averaged across eight benchmarks under BASE and LogicTrack. OA and VU correspond to the coordinates used in Figure 2

Metric	Gemini-2.5		GPT-4o-mini		GPT-5-nano		Llama-3.1-8B		Mistral-7B		Qwen2.5-7B		Qwen2.5-14B	
	Base	LogicTrack	Base	LogicTrack	Base	LogicTrack	Base	LogicTrack	Base	LogicTrack	Base	LogicTrack	Base	LogicTrack
OA ⇑	0.8096	0.8112	0.7068	0.7849	0.6850	0.7044	0.5506	0.5681	0.4964	0.5275	0.6629	0.7060	0.7402	0.7506
VU ⇑	0.1936	0.2725	0.2151	0.4661	0.2607	0.3968	0.0924	0.3275	0.1841	0.2980	0.1633	0.3554	0.2382	0.4143
VWA ⇑	0.3071	0.3693	0.4578	0.6436	0.4527	0.5394	0.3212	0.4717	0.3186	0.4025	0.3507	0.4942	0.4484	0.5573
UR ⇓	0.6273	0.5477	0.3727	0.1898	0.3832	0.2570	0.4171	0.1871	0.3793	0.2476	0.4737	0.3068	0.4213	0.2687

Table 4: Performance of LogicTrack-SFT on each dataset for Llama-3.1-8B. We report OA ↑, VWA ↑, and UR ↓.

Metric	ESNLI	FOLIO	LogiQA	ProntoQA	ProofWriter	QASC	SARA	SemEval
OA	0.6047	0.6121	0.5170	0.9900	0.6548	0.8767	0.6324	0.7842
VWA	0.3366	0.3412	0.2922	0.8069	0.3654	0.5194	0.4029	0.3004
UR	0.4012	0.4735	0.4413	0.1856	0.5069	0.4463	0.3836	0.6207

Table 5: Performance of LogicTrack-SFT on each dataset for Qwen2.5-7B. We report OA ↑, VWA ↑, and UR ↓.

Metric	ESNLI	FOLIO	LogiQA	ProntoQA	ProofWriter	QASC	SARA	SemEval
OA	0.7984	0.6552	0.6136	0.9700	0.6131	0.8767	0.6360	0.7895
VWA	0.4180	0.3329	0.3428	0.6183	0.3302	0.4840	0.4053	0.2982
UR	0.4554	0.4825	0.4313	0.3587	0.5335	0.4589	0.3580	0.6433

Table 6: Performance of LogicTrack-SFT on each dataset for Mistral-7B. We report OA ↑, VWA ↑, and UR ↓.

Metric	ESNLI	FOLIO	LogiQA	ProntoQA	ProofWriter	QASC	SARA	SemEval
OA	0.4651	0.5862	0.4943	0.9800	0.6432	0.8904	0.5184	0.7632
VWA	0.1970	0.3099	0.2846	0.7046	0.3566	0.5057	0.3232	0.2991
UR	0.4864	0.5037	0.4380	0.2934	0.5216	0.4372	0.3958	0.6246

D Prompt Templates

Reasoning Prompt Given sample x, the reasoning model receives a system message instructing it to reason step-by-step in the structured format described in sec:formalization, followed by a user message containing the premises P, query Q, and a dataset-specific answer instruction. The system prompt includes an in-context demonstration and enforces the three-part step format (Premises, Explanation, Derived Fact).

System Prompt for Step-by-Step Reasoning

```
You are a meticulous logician. Read
    user's given premise and hypothesis
    carefully, then reason step-by-step
    explicitly in a Natural Language
    Inference (NLI) style, and conclude
    answer with \boxed{LABEL}.
Do not stop reasoning just because the
    current facts do not prove the
    claim. Stop only when the claim is
    settled, or when no unresolved
    inference path can still settle it.

Each step in the response stage must
    follow the format below, which
    contains three sections: Premises,
    Assumptions, and Derived Fact:

**Response Format**
Step X:
##Premises:##
- [Facts from the input premises or
    prior derived facts.]
- [Don't include any user query
    hypothesis here.]
##Explanation:##
- [Add linguistic bridges only (e.g.,
    synonyms, lexical equivalences,
    definitional mappings)]
- [Never use Assumptions to fill logical
    gaps. Write None if no assumption is
    needed]
##Derived Fact:##
- [The derived facts must be fully
    supported by that step's premises
    and assumptions.]
- [Do not speculate about other facts.]

After your final step, output exactly
    your answer with \boxed{LABEL}.
```

Consistency Check Prompt The consistency check described in sec:verification uses an LLM judge to determine whether the explanations E_i and context C_i conflict with the original given premises P. The judge is instructed to be lenient, only flag direct and unambiguous conflicts.

System Prompt for Consistency Check

```
You are a logical consistency checker.
    Determine whether the assumption in
    the reasoning step conflict with the
    overall context.

*Examples*
- Consistent: given context contains
    'the infant is crying, Tom is
    round', and the assumption says
    'infants are babies; given context
    includes Tom is round'.
- Inconsistent: given context contains
    'dog is round, cat is green', and
    the assumption says 'we don't know
    if cat is round'. Because it
    introduces information not grounded
    in the current context.

Respond with a JSON object in this exact
    format:
{"check_result": "consistent", "reason":
    "..."}
or
{"check_result": "inconsistent",
    "reason": "..."}
```

Answer Extraction The predicted answer $\hat{y}$ is extracted from the model's response via a rule-based procedure. We first look for the `\boxed{...}` notation specified in the prompt, extracting the innermost content and stripping wrappers (e.g., `\text{}`). If no boxed answer is found, we also consider alternative patterns such as `<answer>...</answer>` tags and common answer-declaration phrases (e.g., "the answer is {LABEL}") in the final sentences of the response. If none of these patterns match, the prediction is marked as unanswered.

E The Use of Large Language Models

In this paper, we used GPT-5 [2] and Codex [3] to assist with grammar checking and polishing the writing and LaTeX formatting. The technical ideas, experimental designs, analyses, conclusions, and writing were developed and carried out throughout by the authors. Authors are ultimately responsible for the content of the paper.

[2] https://openai.com/
[3] https://openai.com/codex/