SATBench: Benchmarking LLMs' Logical Reasoning via Automated Puzzle Generation from SAT Formulas

Anjiang Wei[1]*† **Yuheng Wu**[1]* **Yingjia Wan**[2] **Tarun Suresh**[3]
Huanmi Tan[4] **Zhanke Zhou**[1] **Sanmi Koyejo**[1] **Ke Wang**[5] **Alex Aiken**[1]

[1]Stanford University [2]UCLA [3]UIUC [4]CMU [5]Nanjing University

Abstract

We introduce SATBench, a benchmark for evaluating the logical reasoning capabilities of large language models (LLMs) through logical puzzles derived from Boolean satisfiability (SAT) problems. Unlike prior work that focuses on inference rule-based reasoning, which often involves deducing conclusions from a set of premises, our approach leverages the search-based nature of SAT problems, where the objective is to find a solution that fulfills a specified set of logical constraints. Each instance in SATBench is generated from a SAT formula, then translated into a puzzle using LLMs. The generation process is fully automated and allows for adjustable difficulty by varying the number of clauses. All 2100 puzzles are validated through both LLM-based and solver-based consistency checks, with human validation on a subset. Experimental results show that even the strongest model, o4-mini, achieves only 65.0% accuracy on hard UNSAT problems, close to the random baseline of 50%. Our error analysis reveals systematic failures such as satisfiability bias, context inconsistency, and condition omission, highlighting limitations of current LLMs in search-based logical reasoning. Our code and data are publicly available at `https://github.com/Anjiang-Wei/SATBench`

1 Introduction and Method

Logical reasoning is a fundamental component of human intelligence and continues to be a significant challenge in the field of artificial intelligence. The growing interest in the reasoning capabilities of large language models (LLMs) highlights the pressing need for robust benchmarks and evaluation methods [1]. While many datasets have been proposed to evaluate logical reasoning capabilties of LLMs, earlier datasets do not exclusively evaluates logical reasoning in isolution, e.g., LogiQA [2], and ReClor [3], which combine logical reasoning with commonsense reasoning.

Recently, new datasets have been introduced to assess logical reasoning in isolation, such as FOLIO [4] and P-FOLIO [5]. These datasets are manually curated by researchers and focus on logical problems based on *inference rules*, which involve deriving conclusions from a set of premises. A more comprehensive review of related work is provided in Appendix A.

In this work, we introduce SATBench, a benchmark designed to create logical puzzles from Boolean satisfiability (SAT) problems [6, 7] with LLMs. Unlike benchmarks based on inference rules, SAT problems are characterized as *search-based* logical reasoning tasks, where the objective is to determine a truth assignment that fulfills a specified set of logical constraints [8]. This approach to logical reasoning emphasizes a search process akin to backtracking used in SAT solvers. Unlike other

*Equal contribution.
†Correspondence to: anjiang@cs.stanford.edu

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: The 5th Workshop on Mathematical Reasoning and AI.

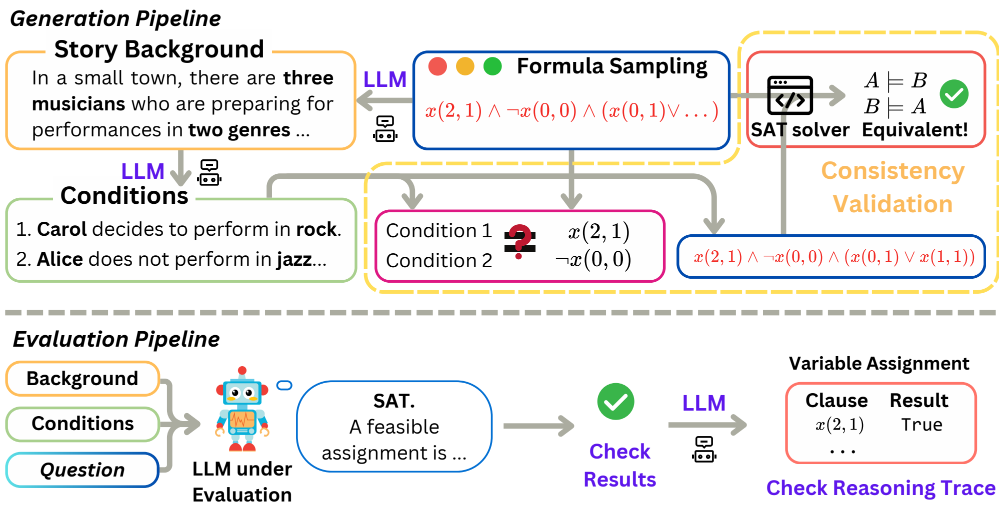

Figure 1: **Overview of the SATBench methodology.** The generation pipeline begins with sampling Conjunctive Normal Form (CNF) formulas, followed by LLM-driven creation of story backgrounds and conditions. To ensure the logical puzzle's quality, both LLM-assisted and solver-based consistency validations are employed. The evaluation pipeline then examines the puzzle's prediction outcomes and checks its reasoning process.

search-based benchmarks such as ZebraLogic [9], which presuppose the existence of a valid solution, SAT problems can result in either a satisfiable solution (SAT) or no solution (UNSAT).

As shown in Figure 1, starting from a SAT formula in Conjunctive Normal Form (CNF), such as $(A \lor \neg B) \land (\neg C \lor \neg D)$, our framework uses LLMs to generate a story context and define a mapping between formula variables and entities in the story. Each clause is then translated into a natural language condition based on this mapping. By sampling CNF formulas with varying numbers of clauses, we can control puzzle difficulty. To ensure the quality of resulting logical puzzles, we reverse the generation process: LLMs translate the natural language conditions back into logical formulas, which are then compared to the originals using a combination of LLM-assisted and solver-based consistency checks. In the evaluation pipeline, we check the result and employ the LLM-as-a-judge strategy to assess the reasoning trace. To validate the overall process of story generation and reasoning trace evaluation, we manually validate 100 examples, increasing confidence in the quality of resulting dataset and evaluation protocol. The detailed construction procedure is provided in Appendix B.

The evaluation on our generated 2100 logical puzzle dataset shows that reasoning models perform well on SATBench, with the o4-mini model achieving the highest accuracy. However, as the number of conditions in the puzzles increases, performance drops noticeably. For the hard UNSAT subset, o4-mini reaches an average accuracy of 65.0%, only slightly above the 50% random baseline. Models perform better on SAT than on UNSAT problems, as UNSAT often requires exhaustive search through the entire solution space. Interestingly, reasoning traces are less reliable for SAT than for UNSAT problems. These results show that SATBench can reveal the limitations of current LLMs in logical reasoning.

Metric	Value
Number of Instances	2100
Average Number of Variables	36.0
Average Number of Clauses	20.6
Average Number of Words	546.2
Average Number of Sentences	55.2

Table 1: Dataset statistics for SATBench.

2 Experiments

2.1 Experimental Setup

Dataset and Prompts. The SATBench dataset contains 2100 logical puzzle instances. Table 1 reports statistics on the average number of Boolean variables and clauses in the sampled SAT formulas,

Table 2: Model accuracy on SATBench using zero-shot prompting for satisfiability prediction. Difficulty levels are categorized as follows: Easy (4-19 clauses), Medium (20-30 clauses), and Hard (31-50 clauses). All open-source models are instruction-tuned.

Model	**SAT** Easy	**SAT** Medium	**SAT** Hard	**UNSAT** Easy	**UNSAT** Medium	**UNSAT** Hard	**Overall** Easy	**Overall** Medium	**Overall** Hard	**Avg.**
Random Baseline	*50.0*	*50.0*	*50.0*	*50.0*	*50.0*	*50.0*	*50.0*	*50.0*	*50.0*	*50.0*
LLaMA3.1-8B	57.9	60.0	48.9	30.4	14.8	17.5	44.1	37.4	33.2	38.2
DeepSeek-Distill-7B	63.9	27.6	16.8	69.1	43.8	42.1	66.5	35.7	29.5	43.9
Qwen3-1.7B	77.1	65.7	53.2	53.4	30.5	42.5	65.3	48.1	47.9	53.7
gpt-4o-mini	82.1	82.4	90.7	42.3	12.9	13.2	62.2	47.6	52.0	53.9
LLaMA4-Scout	84.3	76.7	66.4	52.0	24.3	37.5	68.1	50.5	52.0	56.9
LLaMA3.1-70B	82.0	55.7	45.4	55.2	59.0	48.9	68.6	57.4	47.1	57.7
gpt-4o	85.5	83.3	78.6	54.3	27.1	18.9	69.9	55.2	48.8	58.0
LLaMA3.3-70B	90.7	89.0	75.7	39.5	27.1	30.0	65.1	58.1	52.9	58.7
DeepSeek-Distill-14B	82.9	51.4	41.1	85.7	59.0	51.8	84.3	55.2	46.4	62.0
LLaMA4-Maverick	80.2	86.2	86.1	76.8	25.7	17.9	78.5	56.0	52.0	62.1
Qwen3-4B	84.1	78.1	78.6	80.7	31.9	22.1	82.4	55.0	50.4	62.6
Qwen3-8B	82.7	76.7	67.5	81.6	34.8	32.1	82.1	55.7	49.8	62.6
DeepSeek-Distill-32B	84.5	53.8	42.1	90.0	68.1	58.6	87.2	61.0	50.4	66.2
Qwen3-14B	87.1	72.9	80.0	88.9	47.6	22.1	88.0	60.2	51.1	66.4
Qwen3-235B-Int8	90.0	83.3	83.2	86.1	46.2	19.6	88.0	64.8	51.4	68.1
Qwen-QwQ-32B	92.5	75.7	59.3	84.1	51.9	46.4	88.3	63.8	52.9	68.3
Claude-3.7-Sonnet	88.4	77.6	83.6	93.8	63.3	42.1	91.1	70.5	62.9	74.8
DeepSeek-V3	93.6	83.8	71.4	97.5	83.3	74.3	95.5	83.6	72.9	84.0
DeepSeek-R1	94.8	87.1	73.6	98.2	**89.5**	**83.6**	96.5	88.3	**78.6**	87.8
o4-mini	**97.0**	**96.7**	**91.1**	**98.2**	88.1	65.0	**97.6**	**92.4**	78.0	**89.3**
Average	84.1	73.2	66.7	72.9	46.4	39.3	78.5	59.8	53.0	63.8

as well as the average number of words and sentences in the generated puzzles. The fully automated generation process allows creating more instances if needed. For evaluation, we use 0-shot prompts consisting of a story background, a set of conditions that must be satisfied simultaneously, and a satisfiability query. Models must output a reasoning trace and a final label (`SAT` or `UNSAT`); for SAT, they provide a satisfying assignment, and for UNSAT, an explanation of the conflict. Detailed prompt templates are given in Appendices D.2 and D.3.

Metrics and Models. Satisfiability is framed as a binary classification task (50% random baseline), with accuracy as the main metric. Reasoning trace correctness is judged only when the predicted label is correct, using GPT-4o to verify that the explanation supports the outcome (Appendix B.4). We evaluate proprietary models (GPT-4o, GPT-4o-mini, o4-mini, Claude 3.7 Sonnet) and recent open-source models from the Qwen, Llama, and DeepSeek families. Reasoning trace evaluation focuses on the five best-performing models, with GPT-4o as the judge.

2.2 Main Results

Reasoning models excel, but struggle on harder problems. Table 2 shows zero-shot satisfiability prediction results on SATBench. The best-performing model, o4-mini, reaches 89.3% accuracy, followed by DeepSeek-R1 (87.8%), DeepSeek-V3 (84.0%), Claude 3.7 Sonnet (74.8%), and Qwen-QwQ-32B (68.3%). However, performance drops as difficulty increases: on Hard instances (31–50 clauses), o4-mini falls to 78.0%, and the average accuracy across models is 53.0%, close to the random baseline. Detailed difficulty analysis is provided in Section 2.3.

Scaling Trends. Figure 2 illustrates the scaling trends observed across various model families, such as Qwen3, Llama3.1, Mixtral, Llama4, and DeepSeek-Distill-Qwen. In each family, an increase in model size consistently leads to improved accuracy in satisfiability prediction, thereby validating the anticipated scaling behavior.

2.3 Analysis of Difficulty

SAT versus UNSAT. The "average" row in Table 2 shows a clear accuracy gap between SAT and UNSAT subsets. On Hard instances, models average 66.7% for SAT but only 39.3% for UNSAT,

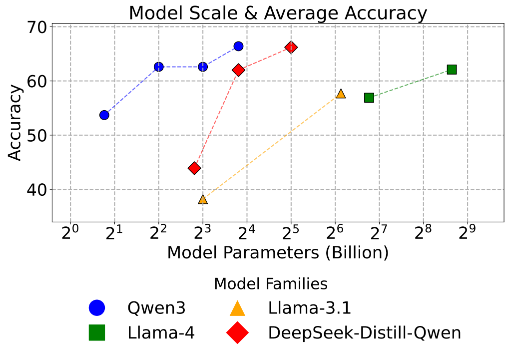

Figure 2: Scaling trend on SATBench.

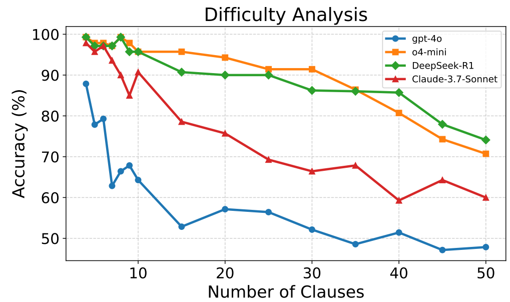

Figure 3: Impact of clause quantity on accuracy.

indicating that SAT is generally easier. This likely stems from the higher complexity of UNSAT problems, which require checking all 2^n assignments for n variables to prove unsatisfiability, while SAT only needs one valid assignment.

Impact of Clause Quantity. Figure 3 shows an inverse relationship between clause count and accuracy. For example, GPT-4o's accuracy drops sharply toward the 50% random baseline as clause count approaches 30. This trend suggests that more clauses increase problem complexity, confirming that our dataset generation method can effectively control difficulty.

2.4 Reasoning Trace Evaluation

We evaluate the reasoning trace validity of various models with GPT-4o, and results are shown in Table 3. The table highlights that DeepSeek-R1 leads in overall trace accuracy with a score of 78.0%, surpassing the o4-mini model by 3.6%. This indicates that while o4-mini excels in prediction accuracy, DeepSeek-R1 provides more reliable reasoning traces.

Model	**SAT** Pred.	**SAT** Trace	**UNSAT** Pred.	**UNSAT** Trace	**Overall Trace**
QwQ	75.5	52.3	60.7	52.4	52.4
Claude-3.7	83.2	47.4	66.4	61.1	54.2
DS-V3	82.9	65.7	85.0	71.1	68.4
o4-mini	94.7	74.6	83.6	74.1	74.4
DS-R1	85.2	73.8	90.3	82.1	78.0

Table 3: Accuracy in prediction and reasoning trace evaluation.

A notable observation is the disparity in trace accuracy between the SAT and UNSAT subsets. Models generally exhibit a more pronounced drop in trace accuracy on SAT problems compared to UNSAT ones. For example, Claude-3.7 experiences a significant 35.8% decrease in trace accuracy on SAT instances, whereas the drop is only 5.3% on UNSAT instances. This pattern indicates that a model's higher prediction accuracy on SAT problems does not necessarily imply it has identified a valid variable assignment. Instead, models exhibit a bias towards predicting SAT outcomes without verifying a valid assignment as evidence.

3 Conclusion

We present SATBench, a benchmark for assessing LLMs' logical reasoning via SAT-derived puzzles. Our dataset features search-based logical reasoning tasks, with controls difficulty and correctness checked by solvers and LLMs. SATBench contains 2100 logical puzzles, and we evaluate both satisfiability prediction and reasoning trace validity. Our findings show model performance drops with increased difficulty, with o4-mini scoring 65.0% on the hard UNSAT cases, near the 50% random baseline. We also conduct an error analysis that identifies systematic patterns such as satisfiability bias, context inconsistency, and condition omission. These findings show that SATBench exposes limitations in current LLMs' ability to perform search-based logical reasoning.

References

[1] M. Luo, S. Kumbhar, M. Shen, M. Parmar, N. Varshney, P. Banerjee, S. Aditya, and C. Baral, "Towards logiglue: A brief survey and A benchmark for analyzing logical reasoning capabilities of language models," *CoRR*, vol. abs/2310.00836, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2310.00836

[2] J. Liu, L. Cui, H. Liu, D. Huang, Y. Wang, and Y. Zhang, "Logiqa: A challenge dataset for machine reading comprehension with logical reasoning," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, C. Bessiere, Ed. ijcai.org, 2020, pp. 3622–3628. [Online]. Available: https://doi.org/10.24963/ijcai.2020/501

[3] W. Yu, Z. Jiang, Y. Dong, and J. Feng, "Reclor: A reading comprehension dataset requiring logical reasoning," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. [Online]. Available: https://openreview.net/forum?id=HJgJtT4tvB

[4] S. Han, H. Schoelkopf, Y. Zhao, Z. Qi, M. Riddell, W. Zhou, J. Coady, D. Peng, Y. Qiao, L. Benson, L. Sun, A. Wardle-Solano, H. Szabó, E. Zubova, M. Burtell, J. Fan, Y. Liu, B. Wong, M. Sailor, A. Ni, L. Nan, J. Kasai, T. Yu, R. Zhang, A. R. Fabbri, W. Kryscinski, S. Yavuz, Y. Liu, X. V. Lin, S. Joty, Y. Zhou, C. Xiong, R. Ying, A. Cohan, and D. Radev, "FOLIO: natural language reasoning with first-order logic," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, Y. Al-Onaizan, M. Bansal, and Y. Chen, Eds. Association for Computational Linguistics, 2024, pp. 22 017–22 031. [Online]. Available: https://aclanthology.org/2024.emnlp-main.1229

[5] S. Han, A. Yu, R. Shen, Z. Qi, M. Riddell, W. Zhou, Y. Qiao, Y. Zhao, S. Yavuz, Y. Liu, S. Joty, Y. Zhou, C. Xiong, D. Radev, R. Ying, and A. Cohan, "P-FOLIO: evaluating and improving logical reasoning with abundant human-written reasoning chains," in *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, Y. Al-Onaizan, M. Bansal, and Y. Chen, Eds. Association for Computational Linguistics, 2024, pp. 16 553–16 565. [Online]. Available: https://aclanthology.org/2024.findings-emnlp.966

[6] S. A. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, ser. STOC '71. New York, NY, USA: Association for Computing Machinery, 1971, p. 151–158. [Online]. Available: https://doi.org/10.1145/800157.805047

[7] L. Pan, V. Ganesh, J. Abernethy, C. Esposo, and W. Lee, "Can transformers reason logically? a study in sat solving," *arXiv preprint arXiv:2410.07432*, 2024.

[8] T. Madusanka, I. Pratt-Hartmann, and R. T. Batista-Navarro, "Natural language satisfiability: Exploring the problem distribution and evaluating transformer-based language models," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 15 278–15 294.

[9] B. Y. Lin, R. L. Bras, K. Richardson, A. Sabharwal, R. Poovendran, P. Clark, and Y. Choi, "Zebralogic: On the scaling limits of llms for logical reasoning," *CoRR*, vol. abs/2502.01100, 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2502.01100

[10] A. Srivastava, A. Rastogi, A. Rao, A. A. M. Shoeb, A. Abid, A. Fisch, A. R. Brown, A. Santoro, A. Gupta, A. Garriga-Alonso *et al.*, "Beyond the imitation game: Quantifying and extrapolating the capabilities of language models," *arXiv preprint arXiv:2206.04615*, 2022.

[11] P. Clark, O. Tafjord, and K. Richardson, "Transformers as soft reasoners over language," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, C. Bessiere, Ed. ijcai.org, 2020, pp. 3882–3890. [Online]. Available: https://doi.org/10.24963/ijcai.2020/537

[12] J. Tian, Y. Li, W. Chen, L. Xiao, H. He, and Y. Jin, "Diagnosing the first-order logical reasoning ability through logicnli," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, M. Moens, X. Huang, L. Specia, and S. W. Yih,

Eds. Association for Computational Linguistics, 2021, pp. 3738–3747. [Online]. Available: https://doi.org/10.18653/v1/2021.emnlp-main.303

[13] J. Jiang, Y. Yan, Y. Liu, Y. Jin, S. Peng, M. Zhang, X. Cai, Y. Cao, L. Gao, and Z. Tang, "Logicpro: Improving complex logical reasoning via program-guided learning," *arXiv preprint arXiv:2409.12929*, 2024.

[14] Q. Zhu, F. Huang, R. Peng, K. Lu, B. Yu, Q. Cheng, X. Qiu, X. Huang, and J. Lin, "Autologi: Automated generation of logic puzzles for evaluating reasoning abilities of large language models," *CoRR*, vol. abs/2502.16906, 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2502.16906

[15] M. Parmar, N. Patel, N. Varshney, M. Nakamura, M. Luo, S. Mashetty, A. Mitra, and C. 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), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, L. Ku, A. Martins, and V. Srikumar, Eds. Association for Computational Linguistics, 2024, pp. 13 679–13 707. [Online]. Available: https://doi.org/10.18653/v1/2024.acl-long.739

[16] Y. Wan, W. Wang, Y. Yang, Y. Yuan, J.-t. Huang, P. He, W. Jiao, and M. R. Lyu, "Logicasker: Evaluating and improving the logical reasoning ability of large language models," 2024. [Online]. Available: https://arxiv.org/abs/2401.00757

[17] D. Sileo, "Scaling synthetic logical reasoning datasets with context-sensitive declarative grammars," 2024. [Online]. Available: https://arxiv.org/abs/2406.11035

[18] N. Patel, M. Kulkarni, M. Parmar, A. Budhiraja, M. Nakamura, N. Varshney, and C. 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, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, Y. Al-Onaizan, M. Bansal, and Y. Chen, Eds. Association for Computational Linguistics, 2024, pp. 20 856–20 879. [Online]. Available: https://aclanthology.org/2024.emnlp-main.1160

[19] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," *arXiv preprint arXiv:1508.05326*, 2015.

[20] A. Talmor, J. Herzig, N. Lourie, and J. Berant, "Commonsenseqa: A question answering challenge targeting commonsense knowledge," *arXiv preprint arXiv:1811.00937*, 2018.

[21] M. Kazemi, Q. Yuan, D. Bhatia, N. Kim, X. Xu, V. Imbrasaite, and D. Ramachandran, "Boardgameqa: A dataset for natural language reasoning with contradictory information," in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., 2023. [Online]. Available: http://papers.nips.cc/paper_files/paper/2023/hash/7adce80e86aa841490e6307109094de5-Abstract-Datasets_and_Benchmarks.html

[22] K. Sinha, S. Sodhani, J. Dong, J. Pineau, and W. L. Hamilton, "CLUTRR: A diagnostic benchmark for inductive reasoning from text," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 4506–4515. [Online]. Available: https://aclanthology.org/D19-1458/

[23] P. Giadikiaroglou, M. Lymperaiou, G. Filandrianos, and G. Stamou, "Puzzle solving using reasoning of large language models: A survey," *arXiv preprint arXiv:2402.11291*, 2024.

[24] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[25] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," *Advances in neural information processing systems*, vol. 36, pp. 11 809–11 822, 2023.

[26] E. Zelikman, Y. Wu, J. Mu, and N. Goodman, "Star: Bootstrapping reasoning with reasoning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 15 476–15 488, 2022.

[27] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," *Advances in neural information processing systems*, vol. 35, pp. 22 199–22 213, 2022.

[28] Y. Li, Z. Lin, S. Zhang, Q. Fu, B. Chen, J.-G. Lou, and W. Chen, "On the advance of making language models better reasoners," *arXiv preprint arXiv:2206.02336*, vol. 2, 2022.

[29] N. Young, Q. Bao, J. Bensemann, and M. Witbrock, "AbductionRules: Training transformers to explain unexpected inputs," in *Findings of the Association for Computational Linguistics: ACL 2022*, S. Muresan, P. Nakov, and A. Villavicencio, Eds. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 218–227. [Online]. Available: https://aclanthology.org/2022.findings-acl.19/

[30] T. Morishita, G. Morio, A. Yamaguchi, and Y. Sogawa, "Learning deductive reasoning from synthetic corpus based on formal logic," in *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, ser. Proceedings of Machine Learning Research, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., vol. 202. PMLR, 2023, pp. 25 254–25 274. [Online]. Available: https://proceedings.mlr.press/v202/morishita23a.html

[31] L. Ranaldi and A. Freitas, "Aligning large and small language models via chain-of-thought reasoning," in *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 1812–1827.

[32] X. Ye, Q. Chen, I. Dillig, and G. Durrett, "Satlm: Satisfiability-aided language models using declarative prompting," *Advances in Neural Information Processing Systems*, vol. 36, pp. 45 548–45 580, 2023.

[33] H. Ryu, G. Kim, H. S. Lee, and E. Yang, "Divide and translate: Compositional first-order logic translation and verification for complex logical reasoning," *arXiv preprint arXiv:2410.08047*, 2024.

Appendix

A Related Work

Table A1: **Comparison of existing logical reasoning benchmarks.** An ideal evaluation framework should meet the following six criteria: (1) Logic Isolation: the benchmark exclusively evaluates logical reasoning in isolation; (2) Automated Generation: the benchmark construction is automated and scalable; (3) Difficulty Control: the difficulty levels of the benchmark questions are adjustable; (4) Natural Language: the questions are written in natural language rather than formal formulas; (5) Template-Free: the benchmark does not rely on expert-designed templates, enhancing diversity; (6) Reasoning Evaluation: the benchmark evaluates both the accuracy of model predictions and the correctness of their reasoning traces.

Benchmark	Search-Based	Logic Isolation	Automated Generation	Difficulty Control	Natural Language	Template-Free	Reasoning Evaluation
LogiQA [2]	✗	✗	✗	✗	✓	✓	✗
BIG-bench [10]	✗	✗	✗	✗	✓	✓	✗
ReClor [3]	✗	✗	✗	✗	✓	✓	✗
RuleTaker [11]	✗	✓	✓	✓	✓	✗	✗
LogicNLI [12]	✗	✓	✓	✗	✓	✓	✗
FOLIO [4]	✗	✓	✗	✗	✓	✓	✗
P-FOLIO [5]	✗	✓	✗	✗	✓	✓	✓
LogicPro [13]	✗	✗	✓	✗	✓	✓	✓
ZebraLogic [9]	✓	✓	✓	✓	✓	✗	✗
AutoLogi [14]	✓	✓	✓	✓	✓	✓	✗
PARAT [7]	✓	✓	✓	✓	✗	✓	✓
LogicBench [15]	✗	✓	✓	✗	✓	✗	✗
LogicAsker [16]	✗	✓	✓	✗	✓	✗	✗
Unigram-FOL [17]	✗	✓	✓	✗	✓	✗	✗
Multi-LogiEval [18]	✗	✓	✓	✓	✓	✗	✗
SATBench (ours)	✓	✓	✓	✓	✓	✓	✓

Logical Reasoning Benchmarks for LLMs Reasoning is a longstanding focus in NLP, with many benchmarks developed to assess model performance. Early efforts targeted natural language inference [19] and commonsense reasoning [20], while recently there has been increasing attention to assessing logical reasoning, as seen in LogiQA [2], ReClor [3], BoardgameQA [21], and CLUTRR [22]. These typically involve reasoning that relies on real-world knowledge. In contrast, datasets like FOLIO [4], RuleTaker [11], and P-FOLIO [5] aim to isolate formal logical reasoning from commonsense knowledge. Logical puzzles have emerged as a compelling testbed in this area [23], with benchmarks including ZebraLogic [9], AutoLogi [14], and LogicNLI [12]. Our work builds on this line by proposing satisfiability-based puzzles [8, 7] for evaluating logical reasoning, using fully automated generation and solver-verified answers. To effectively benchmark the logical reasoning capabilities of LLMs, we propose that an ideal evaluation framework should meet the five criteria, as illustrated in Table A1, while previous works address some of these aspects, few manage to fulfill all these criteria simultaneously.

Logical Reasoning with Language Models Recent work investigates how large language models engage in logical reasoning via prompting techniques, supervised training on reasoning datasets, and translation into formal logic. A prominent line of research focuses on prompting methods that elicit step-by-step reasoning, including chain-of-thought prompting [24], tree-of-thought prompting [25], and self-improvement via bootstrapping [26], along with other methods [27, 28]. Another approach involves fine-tuning LLMs on datasets specifically designed for logical reasoning [29, 30, 1, 31], which has demonstrated improved performance on formal reasoning benchmarks. Complementary to these methods, some work treats LLMs as semantic parsers that convert natural language reasoning tasks into formal logical representations, which are then executed or verified by external solvers or theorem provers [32, 33]. In our evaluation, we use chain-of-thought prompting and prohibit models from invoking external tools; solvers are used only during dataset generation for correctness validation.

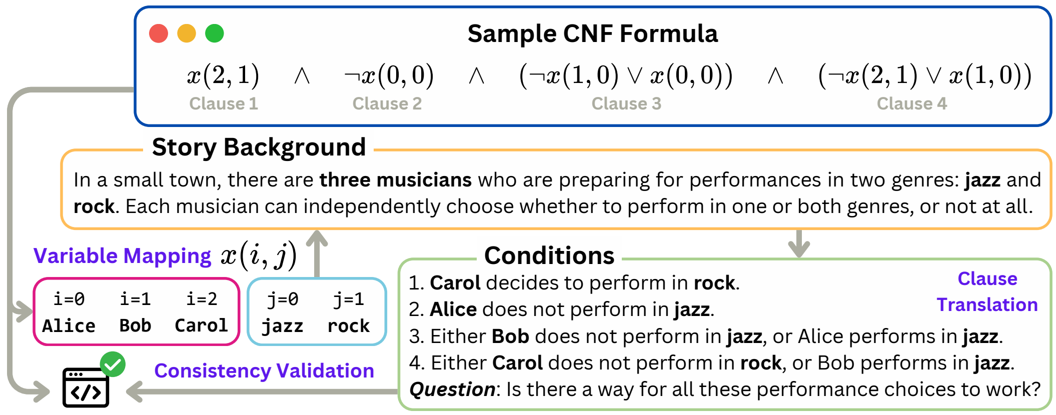

Figure A1: **Benchmark curation pipeline.** The process starts with sampling SAT formulas, followed by using an LLM to generate variable mappings and a story background. Clauses in the formula are then translated into narrative conditions. Consistency between the original formula and the generated puzzle is ensured through both LLM-based and solver-based validation.

B Method

Our objective is to create logical puzzles derived from Boolean satisfiability (SAT) formulas, ensuring the quality of the dataset through both LLM-based and solver-based consistency checks. We further validate each LLM-involved process with human review. The generation method is divided into three stages: SAT formula sampling (Appendix B.1), LLM-based story generation (Appendix B.2), and consistency validation (Appendix B.3). In the evaluation phase, we assess the correctness of the reasoning trace (Appendix B.4).

B.1 SAT formula Sampling

Conjunctive Normal Form (CNF) Conjunctive Normal Form is a structured way of expressing logical formulas, where a formula is a conjunction (AND) of one or more disjunctions (OR) of literals. Each disjunction is referred to as a clause, and each clause consists of literals, which can be either a variable or its negation. For instance, the formula $(x(2,1)) \wedge (\neg x(1,0) \vee x(0,0)) \wedge (\neg x(0,0)) \wedge (\neg x(2,1) \vee x(1,0))$ is in CNF. Here, x represents a two-dimensional array with boolean elements, indicating true or false values. The SAT problem expressed in CNF form involves determining whether there exists an assignment of boolean values to the variables that satisfies the entire formula, making it true. If such an assignment exists, the formula is satisfiable. Conversely, if no such assignment can be found, the formula is unsatisfiable, and an UNSAT-Core can be identified, which is a subset of clauses that are inherently unsatisfiable. This approach constructs puzzles that challenge LLMs to determine if all conditions can be satisfied.

Automation and Difficulty Control The SAT problem can be solved using a SAT solver, which provides a soundness guarantee and allows for an automated and scalable solution. To systematically generate problems with varying levels of difficulty, we can sample formulas that differ in the number of boolean variables and clauses. Additionally, we can increase the dimensionality of the array to create more complex story contexts. By increasing the number of boolean variables, we can generate more clauses to be translated into story conditions. This approach effectively controls the difficulty level by expanding the search space and adding complexity to the constraints, making the search-based logical reasoning more challenging.

B.2 Puzzle Story Generation

Background and Variable Mapping To transform the sampled SAT formula into a narrative context, we utilize a language model, such as GPT-4o, to generate a story background and establish a mapping of variables. For example, as shown in Figure A1, given the SAT formula, the language model creates a scenario involving three musicians: Alice, Bob, and Carol. These musicians are deciding on their performances in two musical genres, jazz and rock. Each musician can independently choose whether to perform in one or both genres, or not at all. The musicians and the genres correspond to the two dimensions of the array x. This mapping is defined as:

$$x(i, j) \to \text{“musician } i \text{ performs in genre } j\text{”}$$

For example:

- $x(0, 0)$: Alice performs in jazz
- $x(1, 0)$: Bob performs in jazz
- $x(2, 1)$: Carol performs in rock

Clause-to-Condition Mapping To transform each clause of the CNF formula into a narrative condition, we employ a large language model (e.g., GPT-4o). This transformation leverages the previously established story background and variable mapping. For example, the clause $\neg x(0, 0)$ is translated to the condition "Alice does not perform in jazz," while the clause $\neg x(2, 1) \vee x(1, 0)$ is expressed as "Either Carol does not perform in rock, or Bob performs in jazz." The final puzzle integrates the story background with these translated conditions and concludes with a question like "Is there a way for all these performance choices to work?" This question serves to assess the satisfiability of the conditions in the logical puzzle.

Our two-phase generation strategy, which begins with the creation of the story background and variable mapping, followed by the transformation of clauses into narrative conditions, improves the tractability and reliability of the process. This structured approach facilitates easier debugging and human validation. Additional examples of generated puzzles are provided in Appendix C.

B.3 Consistency Validation

LLM-based Validation We utilize a large language model (GPT-4o) to ensure that each condition in the generated logical puzzle precisely matches the original SAT formula, given the specified variable mapping. This process checks that no extra conditions are introduced and none are missing. If the check fails, the puzzle is removed from our dataset. The validation prompt is provided in D.1.

Solver-based Validation In addition to LLM-based checks, we adopt solver-based validation that enforces formula-level equivalence between the reconstructed formula and the original CNF. Using the variable mappings, an LLM first converts the narrative conditions back into a SAT formula. The original formula formula A and the reconstructed formula B are checked for equivalence using bidirectional entailment:

$$A \equiv B \quad \text{iff} \quad A \models B \;\wedge\; B \models A.$$

This condition is encoded into a SAT query and checked by the solver. Any reconstructed formula that fails equivalence is discarded, ensuring that the generated puzzles faithfully preserve the original logical structure.

Human Validation To ensure the quality of our dataset, we conduct human validation at two crucial stages involving LLMs, as detailed in Appendix B.2. The first stage involves the generation of the puzzle's background and variable mapping, where humans assess the logical coherence and confirm that the story background accurately reflects the independence of boolean variables. The second stage focuses on the translation of clauses into narrative conditions, where humans ensure that no additional constraints or misinterpretations are introduced.

B.4 Reasoning Trace Evaluation

After generating the logical puzzles, we evaluate an LLM's performance using this dataset. Our evaluation emphasizes both the binary prediction result (SAT or UNSAT) and the validity of the model's reasoning trace. We adopt an LLM-as-a-judge methodology, where the model is instructed to produce a reasoning trace to justify its prediction. Below, we detail the approach for assessing the reasoning trace in SAT and UNSAT scenarios.

SAT Problems When a problem is identified as SAT, it indicates that there is at least one assignment of True or False values to the variables that satisfies the CNF formula. Multiple solutions may exist. For example, consider the CNF formula $(x(0,0) \vee \neg x(1,0)) \wedge (x(1,0) \vee x(2,1))$. One possible satisfying assignment is $x(0,0) = \text{True}$, $x(1,0) = \text{False}$, and $x(2,1) = \text{True}$. After the model predicts a problem as SAT, it is required to generate a reasoning trace to support its prediction. We then instruct the judging LLM to translate this reasoning into a specific variable assignment using the given variable mapping. The judging LLM is further used to verify that each clause in the SAT formula evaluates to True, thereby confirming the satisfiability of the entire SAT formula.

UNSAT Problems Unlike SAT problems, UNSAT problems have no variable assignment that satisfies all clauses. A SAT solver can identify an UNSAT-Core, which is a minimal subset of unsatisfiable clauses. When the model predicts UNSAT, it must provide a reasoning trace.

Consider the formula: $(x(2,1)) \wedge (\neg x(1,0) \vee x(0,0)) \wedge (\neg x(0,0)) \wedge (\neg x(2,1) \vee x(1,0))$. We can demonstrate its unsatisfiability through a step-by-step analysis:

1. From the first clause, $x(2,1)$, we must set $x(2,1)$ to true.
2. From the third clause, $\neg x(0,0)$, we must set $x(0,0)$ to false.
3. Given that $x(0,0)$ is false, the second clause, $\neg x(1,0) \vee x(0,0)$, can only be satisfied if $\neg x(1,0)$ is true, suggesting $x(1,0)$ is false.
4. However, since $x(2,1)$ is true, the fourth clause, $\neg x(2,1) \vee x(1,0)$, can only be satisfied if $x(1,0)$ is true.

This leads to an irreconcilable contradiction: $x(1,0)$ is required to be both true and false simultaneously to satisfy all clauses, rendering the formula unsatisfiable. The example above illustrates a valid reasoning trace for an UNSAT problem in formula format. However, since the model being evaluated lacks access to the variable mapping during its reasoning trace generation, the judging LLM must first translate the reasoning trace back into the variable format. It then compares this translated reasoning with the provided UNSAT-Core to assess the accuracy of the reasoning trace.

Human Validation Given our use of an LLM-as-a-judge methodology for evaluating reasoning traces, we incorporate a human validation process to check the correctness of the LLM's judgments.

C Example of Generated Puzzles

Please see Figure A2.

D Templates

D.1 LLM Validation Prompt Template

Please see Figure A3.

D.2 SAT/UNSAT Evaluation Prompt Template

Please see Figure A4.

D.3 Trace Evaluation Prompt Template

Please see Figure A5 and Figure A6.

E Error Analysis Examples

We show one representative example for each of the four error types, paraphrased for clarity.

Satisfiability Bias The model outputs an assignment such as $x(1) = 1, x(2) = 1, x(3) = 0$ and prematurely declares the formula satisfiable. In reality, satisfiability requires that *all* clauses be satisfied, yet several clauses remain violated. This suggests that the model often assumes satisfiability without exhaustively checking all constraints and fails to engage in search-based logical reasoning with backtracking.

Context Inconsistency The model produces conflicting assignments for the same variable within one trace. For example, it first sets $x(0) = 1$ but later assigns $x(0) = 0$, trying to satisfy $x(0) \wedge \neg x(0)$. This is impossible: a variable can only take a single value. The correct resolution is either to retain one consistent assignment in a satisfiable case or to conclude UNSAT when no such assignment exists.

Condition Omission The model may ignore or hallucinate conditions in its reasoning trace. For example, given $(x(0) \vee x(1)) \wedge \neg x(0)$, it incorrectly reduces the formula to $x(0) \wedge \neg x(0)$, which is unsatisfiable. In reality, the original formula is satisfiable with $x(0) = 0, x(1) = 1$. Such omissions cause the model to misclassify satisfiable instances as UNSAT.

Spurious Priors The model introduces commonsense assumptions that are absent from the formula. For example, with $x(0) \vee x(1)$, the model assumes that $x(0)$ and $x(1)$ cannot both be true (as if they were "mutually exclusive"). It then treats the assignment $x(0) = 1, x(1) = 1$ as a contradiction and concludes UNSAT. In reality, the formula is satisfiable and permits both variables to be true. This is because models sometimes introduce commonsense assumptions that are absent from the given constraints.

F Human Validation

We conducted human validation on a uniformly random sample of 100 puzzles from our generated dataset to verify the correctness of LLM-involved steps and the reliability of our evaluation protocol. Each puzzle contains a CNF formula, its satisfiability label (SAT or UNSAT) from a symbolic solver, a narrative scenario with variable mappings, natural language conditions corresponding to each clause, a reasoning trace generated by an LLM, and an LLM judgment of whether the reasoning trace is logically valid. Three co-authors independently annotated the sample and resolved disagreements by majority vote.

Annotators performed three validation tasks:

1. **Scenario and Mapping Consistency:** Ensuring that all entities in the scenario are covered in the variable mapping, and that every logical variable is correctly grounded. We observed no errors (100% accuracy).
2. **Clause Translation Faithfulness:** Verifying that each clause in the CNF formula is faithfully translated into its natural language condition without omissions, additions, or misinterpretations. We found minor translation errors in three cases, yielding a 97% accuracy rate.
3. **LLM Judgment Correctness:** Checking whether the LLM's judgment of the reasoning trace is logically correct and aligned with the ground-truth formula and satisfiability label. Here, accuracy was 93%, with occasional errors due to incomplete assignment extraction or overly strict interpretations of valid traces.

Overall, these results confirm the robustness of our dataset and evaluation pipeline, with errors being rare and not significantly affecting reliability.

A few failure cases were observed. In story generation, one error involved the clause $(\neg x(2, 0) \vee x(2, 1))$ being translated as "if Dr. Brown is not assigned project 0, then Dr. Brown is assigned project 1." This misuses the *if-then* structure. The correct phrasing is "if Dr. Brown is assigned project 0, then Dr. Brown is also assigned project 1."

For the LLM-as-judge setting, the main error mode involved incomplete extraction of the assignment within the trace. In some cases, the model judged that the trace was invalid, even though the trace was logically sound. These minor errors, however, were rare and did not affect the overall robustness of our pipeline.

Puzzle Example

```
<SAT formula>
(¬x(0, 0) ∨ x(1, 0)) ∧ (x(0, 1)) ∧ (¬x(1, 0)) ∧ (¬x(0, 1) ∨ x(1,
    1))
∧ (¬x(1, 1) ∨ x(0, 0))
</SAT formula>

<satisfiable>
false
</satisfiable>

<UNSAT reason>
Frozen conflict chain: sequential forced assignments leading to
    contradiction:
(x(0, 1)), (¬x(0, 1) ∨ x(1, 1)), (¬x(1, 1) ∨ x(0, 0)),
(¬x(0, 0) ∨ x(1, 0)), (¬x(1, 0))
</UNSAT reason>

<scenario>
Two wildlife researchers, Hannah and Liam, are documenting animal
    behavior at a sanctuary. They are independently recording
    whether they observe two specific behaviors: feeding (0) and
    social interaction (1). Each researcher decides on their own
    which behavior they have observed, and they may report multiple
    behaviors or none at all.
</scenario>

<variable_mapping>
Let x(i, j) mean researcher i observes behavior j.
Here, researcher 0 is Hannah, and researcher 1 is Liam.
</variable_mapping>

<conditions>
1. Either Hannah does not observe feeding, or Liam observes feeding.
2. Hannah observes social interaction.
3. Liam does not observe feeding.
4. Either Hannah does not observe social interaction, or Liam
    observes social interaction.
5. Either Liam does not observe social interaction, or Hannah
    observes feeding.
</conditions>

<question>
Is there a way to assign observations that make this work?
</question>
```

Figure A2: Puzzle Example.

LLM Validation Prompt Template

```
You are a logic checker.

You are given a SAT formula, a variable explanation, and a natural
    language puzzle based on the formula. Your job is to check
    whether the natural language conditions are logically equivalent
     to the original SAT formula.

Specifically, for each clause in the SAT formula:
- Verify there is a corresponding natural language condition with
    equivalent logical meaning.
- Ensure the variable usage matches the explanation format.
- Make sure there are no missing clauses, no added constraints, and
    no changes in logic.

Pay special attention to logical implications and how they are
    expressed in natural language. For example:

The clause (¬x(2) ∨ x(1)) is logically equivalent to: ``If x(2) is
    true, then x(1) is also true." A common mistake is to write this
     as: ``If ¬x(2) then x(1)", which is incorrect. That corresponds
     to the clause (x(2) ∨ x(1)), and changes the meaning.

Here is the information:

<scenario>
{scenario}

<variable explanation>
{variable_mapping}

<conditions>
{conditions}

<question>
{question}

<SAT formula>
{formula}

Think step by step about whether the SAT formula and the natural
    language conditions match logically, clause by clause. Consider
    the number of clauses, the variable usage, and the logical
    operators involved.

Your job is only to evaluate whether each condition correctly
    represents its corresponding clause in the SAT formula. You
    should not judge whether the overall formula or the scenario is
    satisfiable, solvable, or logically consistent.

Do not attempt to rewrite, fix, or invent any missing conditions. If
     any clause is missing, mistranslated, or not clearly
    represented, you must mark the result as [INCONSISTENT].

Finally, in the last line, output either [CONSISTENT] or [
    INCONSISTENT].
Do not include anything after this label.
```

Figure A3: LLM Validation Prompt Template.

SAT/UNSAT Evaluation Prompt Template

```
You are a logical reasoning assistant. You are given a logic puzzle.

<scenario>
{scenario}

<conditions>
{conditions}

<question>
{question}

Guidelines:
- All constraints come **only** from the <conditions> section.
- The <scenario> provides background and intuition, but **does not
   impose any additional rules or constraints**.
- All variables represent **independent decisions**; there is no
   mutual exclusivity or implicit linkage unless stated explicitly
   in <conditions>.
- Variables not mentioned in <conditions> are considered unknown and
    irrelevant to satisfiability.

Your task:
- If the puzzle is satisfiable, propose one valid assignment that
   satisfies all the conditions.
- If the puzzle is unsatisfiable, explain why some of the conditions
    cannot all be true at once.

Think step by step. At the end of your answer, output exactly one of
    the following labels on a new line:
[SAT] - if a valid assignment exists
[UNSAT] - if the constraints cannot be satisfied

Do not add any text or formatting after the final label.
```

Figure A4: SAT/UNSAT Evaluation Prompt Template.

Trace Evaluation Prompt Template for SAT Prediction

```
You are given a logical puzzle and a reasoning trace from a language
     model.

The puzzle is also expressed as a SAT formula. Each clause is a
    disjunction (OR) of literals formatted like x(i), x(i,j), or x(i
    ,j,k). These variables follow the meaning:

- x(i) means object or person i has some unnamed property.
- x(i,j) means object i has property or role j.
- x(i,j,k) means object i has property j in context or slot k (e.g.,
     time, situation, location).

A positive literal like x(0,1) means that the property is present.
A negative literal like ¬x(0,1) means it is absent.

Below is the full logical puzzle and its corresponding formula:

<scenario>
{scenario}

<conditions>
{conditions}

<final question>
{question}

<variable explanation>
{variable_mapping}

<SAT formula>
{formula}

<trace from model>
{model_trace}

Your task is to extract the truth assignment implied by the model's
    reasoning trace, and evaluate whether each clause in the SAT
    formula is satisfied.

Go through the trace and determine whether each variable appearing
    in the SAT formula is marked as True or False.

Then, for each clause, evaluate the truth value of each literal
    using this assignment.

For example, if a clause in the SAT formula is (x(0) ∨ ¬ x(1)), and
     the model says x(0) is True and x(1) is also True, then this
    clause becomes [1, 0].

Think step by step. Show the variable assignments and how you
    evaluate each clause.

Finally, in the **last line**, output a single line in the format:
Assignment: [[1, 0], [0, 1, 1], [1], ...]

For any variable that is not explicitly mentioned in the reasoning
    trace, assume its value is 0 when constructing the assignment
    list. Do not include anything after this label.
```

Figure A5: Trace Evaluation Prompt Template for SAT Prediction.

Trace Evaluation Prompt Template for UNSAT Prediction

```
You are evaluating whether a model's reasoning trace correctly
    explains an UNSAT logical puzzle.

<scenario>
{scenario}

<conditions>
{conditions}

<question>
{question}

<variable explanation>
{variable_mapping}

<reasoning trace from model>
{model_trace}

<ground-truth unsat reason>
{unsat_reason}

We already know this puzzle is UNSAT (unsatisfiable).
Your task is to judge whether the reasoning trace correctly
    identifies or meaningfully reflects the cause of
    unsatisfiability - that is, whether it aligns with the given
    ground-truth unsat reason, even if it doesn't name it explicitly
    .

Focus on logical precision:
- Does the trace show or imply a variable assignment or chain of
    reasoning that leads to contradiction?
- Does it avoid hallucinations or irrelevant claims?

Note: The trace may present a specific variable assignment or
    reasoning path that leads to a contradiction. Whether it aligns
    with the given ground-truth UNSAT reason means you must judge
    whether the contradiction is logically valid and reflective of
    the actual cause, even if it doesn't explicitly name the minimal
     core or unsat pattern.

You are **not** evaluating whether the conclusion "UNSAT" is correct
     - that is already known to be correct.
You are only evaluating whether the explanation substantively
    captures why the instance is unsatisfiable.

Please think step by step. First, explain whether and how the
    reasoning trace aligns with the unsat reason.
Then, in the last line, output one of the following labels:

[YES] - the reasoning trace is logically valid and correctly
    captures the UNSAT cause
[NO] - the trace is flawed, incomplete, or does not match the
    correct unsat reason

Do not include anything after this label.
```

Figure A6: Trace Evaluation Prompt Template for UNSAT Prediction.