There's No Such Thing as Simple Reasoning for LLMs

Nurul Fajrin Ariyani, Zied Bouraoui, Richard Booth, Steven Schockaert

There's No Such Thing as Simple Reasoning for LLMs

Nurul Fajrin Ariyani[1], Zied Bouraoui[2], Richard Booth[1], Steven Schockaert[1]
[1]Cardiff NLP, Cardiff University, UK [2]CRIL-CNRS & University of Artois, France
{ariyaninf,boothr2,schockaerts1}@cardiff.ac.uk bouraoui@cril.fr

Abstract

Large Language Models (LLMs) have been widely found to struggle with logical reasoning, where even fine-tuned models fail dramatically on out-of-distribution problems. However, existing work has focused on relatively complex "many-hop" reasoning problems. In this paper, we analyse the performance of fine-tuned LLMs on simple reasoning problems, all of which can be solved in at most three inference steps. Due to the simplicity of these problems, the model cannot encounter test problems that are fundamentally different from those it has seen during training. Unfortunately, however, we find that the models remain highly brittle, being susceptible to seemingly innocent perturbations, such as the addition of duplicates to the set of premises and shuffling the order in which the premises are presented.

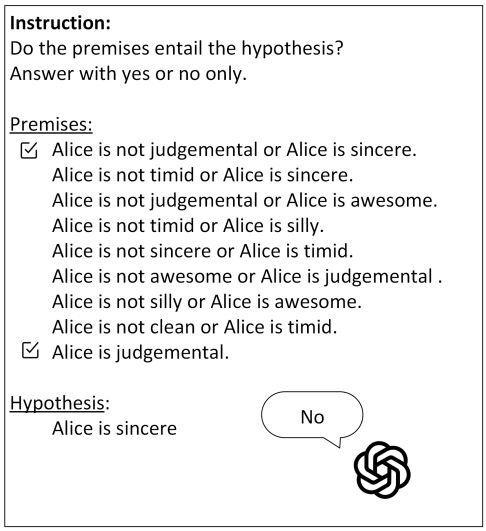

Figure 1: A simple reasoning problem where the hypothesis can be inferred by combining two premises. However, GPT-4o fails to answer correctly.

1 Introduction

In its simplest form, logical reasoning involves inferring logical consequences by combining the information that is encoded in a given set of premises. Consider, for example, the following premises: (i) *Alice is not judgmental or Alice is sincere*; (ii) *Alice is judgmental or Alice is timid*; (iii) *Alice is not timid*. Premises (i) and (ii) imply: (iv) *Alice is sincere or Alice is timid*. Combining (iv) and (iii), we can furthermore derive: (v) *Alice is sincere*. In this example, deriving (v) required two derivation steps. In general, logical reasoning problems may involve a large number of such derivation steps. Several recent studies have shown that the logical reasoning abilities of Large Language Models (LLMs) are surprisingly limited (Patel et al., 2024; Tian et al., 2021; Wang et al., 2023; Parmar et al., 2024). While it is possible to fine-tune LLMs to improve their reasoning abilities, the resulting models struggle with out-of-distribution test examples and they are typically sensitive to language variance (Zhang et al., 2023; Chang et al., 2024; Mirzadeh et al., 2024).

However, previous evaluations were conducted on fairly complex reasoning tasks (Chen et al., 2023; Wang et al., 2023; Dalvi et al., 2021), where e.g. up to 10 derivation steps have to be combined to infer a given conclusion. Such reasoning problems rarely arise in everyday discourse. We therefore consider the following research question: can LLMs be fine-tuned to reliably carry out *simple* logical reasoning tasks? We specifically consider reasoning problems that require at most 3 inference steps. Such problems can typically be solved by humans at a glance. Given the impressive capabilities of recent LLMs, we would thus expect them to be capable of solving such problems as well. Furthermore, while we focus on synthetically generated examples in our analysis, the ability to solve simple reasoning problems is clearly of practical significance. For instance, such forms of reasoning are commonly needed for resolving ambiguities in text. Similarly, RAG systems may need to combine

information from different sources (Jeong et al., 2024; Lazaridou et al., 2022).

Unfortunately, even for simple reasoning problems, we find that LLMs perform surprisingly poorly. To illustrate this, Figure 1 shows a simple reasoning problem on which GPT-4o failed, despite the fact that the hypothesis can be derived in a single inference step. Motivated by this observation, we focus our analysis in this paper on fine-tuned models. Consistent with previous work, we find that fine-tuned models can solve reasoning problems with near-perfect accuracy when test examples are generated in the same way as training examples. Moreover, due to the simplicity of the considered reasoning problems, it is not possible for the model to encounter test examples that have a different structure than those it has seen in training. Nonetheless, we find that the performance of fine-tuned LLMs deteriorates significantly on out-of-distribution examples. A closer analysis reveals that models struggle with seemingly innocent differences in how examples are presented. For instance, we find that the models struggle when premises in the input are duplicated. Furthermore, they are also highly sensitive to the order in which the premises are presented, with the position of literals being particularly important. These findings show that even for simple reasoning problems, fine-tuned LLMs rely on shortcuts rather than genuinely learning to reason, which is in accordance with what has been found for other forms of reasoning (Nikankin et al., 2024).

2 Related Work

Evaluating the reasoning abilities of LLMs has been the subject of an extensive line of work, which has covered, among others, deductive (Clark et al., 2020; Saeed et al., 2021; Parmar et al., 2024; An et al., 2024), abductive (Kazemi et al., 2023; Tafjord et al., 2021), commonsense (Tian et al., 2021; Dalvi et al., 2021), and symbolic reasoning (Pan et al., 2023; Jiang et al., 2024). A common finding is that the reasoning depth, i.e. the number of inference steps that need to be chained to arrive at the answer, is a strong predictor of problem difficulty (Dziri et al., 2023; Parmar et al., 2024). In particular, even fine-tuned LLMs fail to generalize to problems that require longer inference chains than those in the training set (Zhang et al., 2023). Chain-of-thought prompting (Kojima et al., 2022; Saparov and He, 2023) and providing in-context demonstrations (Wang et al., 2023) have been found to improve reasoning abilities. However, such techniques are less effective for smaller models, and they come with their own limitations (Wei et al., 2022; Stechly et al., 2023, 2024). Reasoning models, such as OpenAI o1 and DeepSeek R1, can overcome some of these limitations, but at the expense of a significant computational overhead (Valmeekam et al., 2024). A practical solution is to use LLMs to convert problem instances to symbolic inputs for external solvers, and thus avoid relying on the LLM to solve the reasoning problems themselves (Jiang et al., 2024; Pan et al., 2023). While this may indeed be desirable for complex reasoning problems, we would still expect LLMs to be capable of making simple logical inferences, as this capability may be needed for general language understanding (e.g. for resolving ambiguities).

To the best of our knowledge, only few works have focused on simple reasoning problems. Yang et al. (2024) showed that LLMs can sometimes fail on simple problems, even if they perform well on harder ones. A study on simple math reasoning tasks involving two or three operands further underscores the sensitivity of LLMs when exposed to variations in numerical and textual framing (Stolfo et al., 2023). LogicAsker (Wan et al., 2024) evaluates the basic reasoning skills of LLMs, but their focus is on off-the-shelf models such as GPT-4 and Gemini 1.5, whereas we focus on the potential of fine-tuning models on simple reasoning tasks.

3 Experimental Setup

The problems we consider involve a set of premises and a hypothesis, all of which are natural language verbalizations of formulas from propositional logic. The task is to predict whether the hypothesis is logically entailed by the premises (w.r.t. the usual semantics of propositional logic). An example of a problem instance is shown in Figure 1. The verbalizations are obtained using simple templates, which makes the statements easy to parse and allows us to focus specifically on the reasoning abilities.

Training Data To fine-tune our LLMs, we use a dataset that is obtained as follows.[1] We first randomly sample a set of clauses $\{\phi_1, ..., \phi_m\}$. Each clause is a literal or a disjunction of two literals, where a literal is either an atomic proposition or the negation of an atomic proposition. Each time

[1]Our code is available at https://github.com/ariyaninf/simple_reasoning.

a clause is added to the set, we use a SAT solver (Le Berre and Parrain, 2010) to check that it remains satisfiable; if not then the clause is discarded and we repeat the sampling process. After the full set of clauses has been sampled, we randomly sample one literal as the hypothesis. Further details on the sampling algorithms are explained in Appendix A.

Note that these problem instances are particularly straightforward. For instance, because at most two literals appear in each clause, they fall within the polynomial fragment of propositional logic. We further restrict the difficulty of the problem instances by avoiding problems with an inference depth of more than 3. Specifically, for a positive instance, where the hypothesis is logically entailed by the premises, we define the inference depth as the number of inference steps which are needed to derive the conclusion. An inference depth of k means that the hypothesis is entailed by a subset of $k+1$ premises. Inference depth 0 thus means that the hypothesis is included in the set of premises. For negative examples, the inference depth is defined based on the immediate consequence operator. Specifically, starting from the given set of premises, the immediate consequence operator adds all premises that can be derived in one step. The inference depth of a negative problem instance is defined as the number of times that the immediate consequence operator can be applied before the set of premises saturates. For instance, an inference depth of 0 means that all clauses that are logically entailed by the set of premises are already included in the set. Intuitively, problems with a higher inference depth are more challenging.

We ensure that the training data is balanced in two ways. First, we ensure that there is an equal number of problems of depths 0, 1, 2 and 3. Second, for each inference depth, we ensure that there is an equal number of positive and negative examples. The training set consists of a total of 100K problem instances, where the number of atomic propositions in each problem instance varies from 2 to 15, and the number of premises from 2 to 50. A further 10K problem instances are sampled for validation.

Test Data We experiment with three different test sets. First, we sample a set of 1000 problem instances following the same process as for the training data. We will refer to this test set as SAT. To test the robustness of the model to out-of-distribution instances, we also generate test sets following the Rule Priority (RP) and Label Priority (LP) sampling strategies that were proposed by Zhang et al. (2023). For both strategies, we sample 1000 problem instances. One key difference with SAT is that RP and LP only include Horn clauses. Specifically, the clauses in these two test sets are either positive literals (i.e. atomic propositions), or clauses of the form $\neg p \vee q$, where p and q are atomic propositions.

4 Results and Analysis

Table 1 summarizes the performance of a number of standard LLMs. Although our main focus is on evaluating the performance of fine-tuned models, to put the results in context, we also include results for three pre-trained models, which are evaluated in a zero-shot fashion: GPT-4o, o3-mini, and DeepSeek-R1-Distill-Llama-8B. In the case of GPT-4o, we experimented both with a standard prompt and with a chain-of-thought prompt (see Appendix B.2). The fine-tuned models were trained on 100K problem instances, obtained using the SAT sampling strategy, as explained in Section 3.

A number of clear observations can be made. First, GPT-4o and the distilled DeepSeek-R1 model significantly underperform the fine-tuned models. In the case of GPT-4o, using CoT prompting somewhat improves results, although the overall performance remains surprisingly disappointing, given the simplicity of the considered problem instances. For example, even in cases where the hypothesis is listed among the premises (i.e., $k = 0$), the model still make mistakes in a non-trivial number of cases. Furthermore, for the LP dataset with $k = 3$, the results for GPT-4o with standard prompting and the distilled DeepSeek-R1 model are both around random guessing (i.e., 50%). As a more powerful closed-source model, o3-mini performs better, although it still makes mistakes, even in the simplest settings. We noticed a slight improvement in o3-mini with CoT, particularly for problems with a higher depth. We discuss the performance of o3-mini further in Appendix C.2.

The fine-tuned models achieve perfect accuracy for $k = 0$. However, for $k = 3$, we see a clear deterioration in performance. Comparing the different models, we can see that the smallest models (Llama-3.2 and Phi-4-mini) underperform, while Llama-3.1 and Ministral-8B generally perform best. It is notable that all fine-tuned models perform considerably worse on RP and LP. While previous work has already highlighted the fact that fine-

	SAT				RP				LP			
	k=0	k=1	k=2	k=3	k=0	k=1	k=2	k=3	k=0	k=1	k=2	k=3
Pre-trained models												
GPT-4o (standard)	93.2	83.6	80.0	76.8	93.6	67.6	58.0	57.2	91.6	64.8	51.6	51.2
GPT-4o (CoT)	97.2	90.0	82.0	72.0	100	81.6	66.0	53.2	99.6	84.8	80.4	65.6
o3-mini (standard)	100	99.2	98.4	98.0	100	99.6	100	98.8	99.2	100	100	98.4
o3-mini (CoT)	100	100	99.6	99.6	100	99.6	100	99.6	99.2	100	100	99.6
DeepSeek-R1-Distill-Llama-8B (COT)	92.0	75.2	65.2	63.6	87.2	73.6	61.2	56.4	84.4	71.2	55.2	47.6
Fine-tuned models												
Meta-Llama-3-8B-Instruct	100	99.2	99.6	97.2	100	99.6	99.2	94.0	100	99.6	99.2	94.0
Llama-3.1-8B-Instruct	100	99.2	99.6	97.6	100	99.6	99.2	97.2	100	99.6	98.8	98.0
Llama-3.2-3B-Instruct	100	99.6	99.6	97.2	100	95.6	92.8	85.6	100	87.2	76.4	68.8
Mistral-7B-Instruct-v0.3	100	99.6	97.6	98.4	100	99.6	96.4	93.2	100	98.4	89.6	85.6
Ministral-8B-Instruct-2410	100	98.8	100	98.4	100	99.2	99.2	98.0	100	98.8	98.0	97.2
Qwen2.5-7B-Instruct	100	99.2	99.6	96.4	100	98.8	98.4	94.0	100	95.4	83.7	73.3
Phi-4-mini-insruct	100	98.0	97.6	94.4	100	96.4	94.4	90.0	100	95.6	84.4	78.8
gemma-3-4b-it	100	100	98.4	97.6	100	100	97.6	90.0	100	98.0	90.8	78.0

Table 1: Results for the SAT, RP and LP test sets, for different inference depth k (accuracy).

tuned models can struggle on out-of-distribution examples (Zhang et al., 2023), this result is still somewhat unexpected. Indeed, due to the simplicity of the training and test problems, it is not immediately clear that the RP and LP test examples can really be considered to be out-of-distribution. However, upon closer inspection, we identified two aspects in which these datasets differ. First, in the case of LP, some of the premises are duplicated in the input. Second, in the case of RP and LP, all literals (i.e. clauses of length 1) appear at the end of the list of premises, whereas for SAT they appear in between other premises. Despite the seemingly innocent nature of these differences, we found them to have a clear impact on performance, as we show in the remainder of this section.

Sensitivity to Duplicated Premises Figure 2 shows the results of Llama-3.1-8B-Instruct on a number of variants of the test sets, where up to 10 premises have been duplicated. To create these variants, we repeatedly select one premise from the list and add a copy of that premise at the end of the list.[2] For RP and LP we can see a clear drop in performance as a result of the added duplicates.

Sensitivity to Literal Positioning Figure 3 shows results on some variants of the test sets, where the position of the literals is changed: *none* refers to the original dataset, *end* is a variant where all literals are placed at the end, *shuffled* is a variant where all premises are shuffled, and literals thus

[2]Note that the same premise might be duplicated more than once using this strategy.

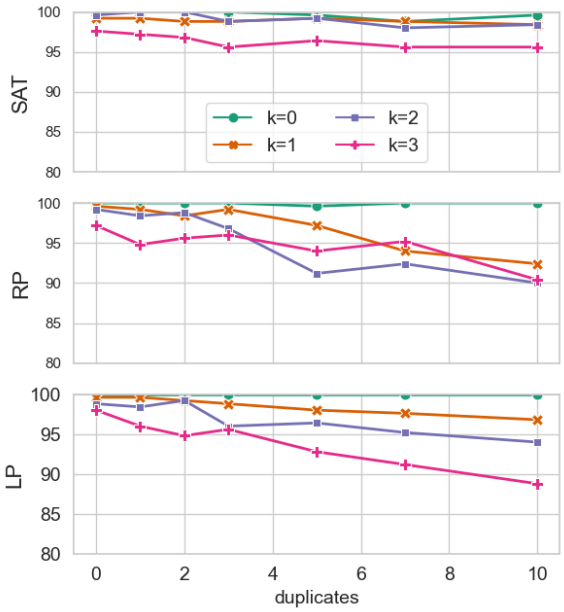

Figure 2: Accuracy for Llama-3.1-8B-Instruct when varying numbers of duplicates are added to premises in test instances

.

appear at random positions, and *first* is a variant where all literals are placed at the start. The results show that *first* leads to the worst results, with drops of up to 14% in the case of RP. For LP and RP, the *end* and *none* variants coincide. Interestingly, for SAT, we can see a difference in performance between *none* and *shuffled*, which shows that the SAT sampling process introduces a bias in the position of literals, due to the use of consistency checks after each clause is sampled.

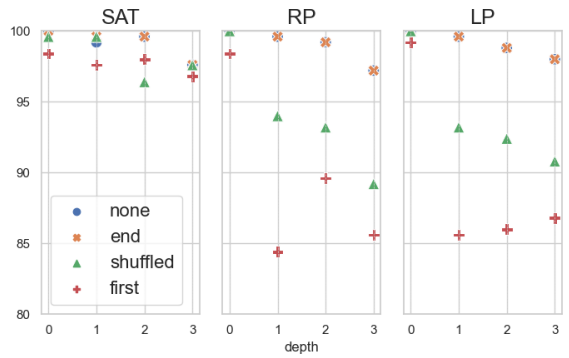

Figure 3: Accuracy for Llama-3.1-8B-Instruct when varying the position of literals in test instances.

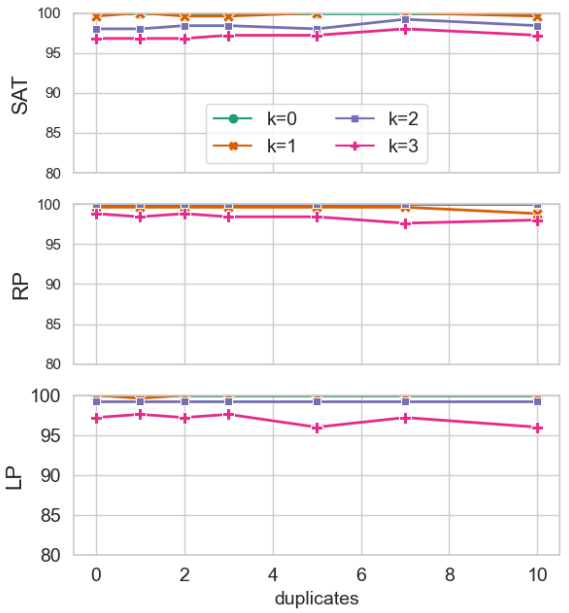

Figure 4: Accuracy for Llama-3.1-8B-Instruct when we introduced duplicated premises in training examples and tested it with the same test sets as in Fig. 2.

Mitigating the Sensitivity Issues One may wonder whether the two aforementioned sensitivity issues can be addressed by modifying the training data. To test this, we first consider an experiment where the model is fine-tuned on training examples with duplicated premises. Specifically, for each of the SAT training examples we added a number of duplicated premises, sampled uniformly between 0 and 10. Figure 4 shows improved results, for a fine-tuned Llama-3.1-8B-Instruct model, compared to Figure 2, although introducing duplicates still has a negative effect for LP.

We used a similar strategy to address the literal positioning issue. In this case, we modified the SAT training examples by altering the position of

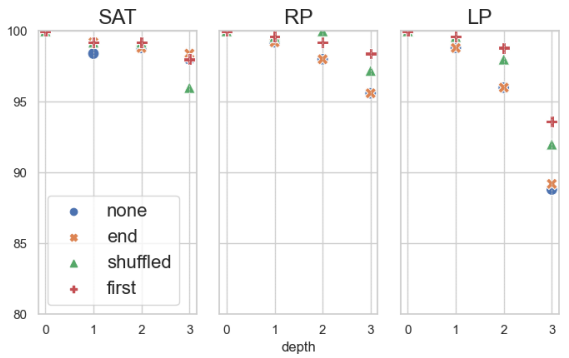

Figure 5: Accuracy for Llama-3.1-8B-Instruct fine-tuned on SAT with varied literal positioning and tested using the same test sets as in Fig. 3

.

literals. To keep the training data balanced, for every inference depth, we maintained an equal representation of the four considered literal positioning variations: none, shuffled, end, and first. We enforce this separately for positive and negative examples. As before, the dataset consists of 100K training problems and 10K validation problems. The results in Figure 5 show that this modification to the training data can reduce the performance gap between positioning variations at each inference depth. However, the model still struggles with out-of-distribution test sets, as the performance on LP with $k = 3$ is surprisingly worse than in Figure 3.

5 Conclusions

This paper investigated the ability of LLMs to reliably and consistently perform simple reasoning. We found that even on problems that only require a single reasoning step, fine-tuned LLMs make some mistakes. Moreover, while such mistakes are rare on in-distribution examples, we found that the performance can deteriorate dramatically when seemingly innocent changes are made to test problems. In particular, we highlighted the effect of duplicated premises and changing the order where literals are positioned. While it is possible to alleviate the specific issues that we identified by adapting the training data, the fact that these issues arose in the first place strongly suggests that fine-tuned models rely on reasoning shortcuts, even for the very simple reasoning problems that we have considered. As such, model performance should be expected to deteriorate whenever out-of-distribution problem instances are encountered.

Limitations

For the fine-tuned models that we analysed in this paper, we have relied on simple prompts, asking the model to directly provide the answer. Strategies such as chain-of-thought prompting may lead to more robust results, especially when combined with reinforcement learning. For instance, while the results with the distilled DeepSeek-R1 model were disappointing, fine-tuning such a model may lead to more robust results. One challenge, however, is that chain-of-thought derivations cannot straightforwardly be used to prove that a premise cannot be entailed.

Acknowledgments

Nurul Ariyani was supported by the Center for Higher Education Funding and Assessment, the Ministry of Higher Education, Science, and Technology of Republic Indonesia, and the Indonesia Endowment Fund for Education (LPDP). Steven Schockaert was supported by EPSRC grant EP/W003309/1. Zied Bouraoui was supported by ANR-22-CE23-0002 ERIANA.

References

Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, Dong Chen, Dongdong Chen, Junkun Chen, Weizhu Chen, Yen-Chun Chen, Yi-ling Chen, Qi Dai, Xiyang Dai, Ruchao Fan, Mei Gao, Min Gao, Amit Garg, Abhishek Goswami, Junheng Hao, Amr Hendy, Yuxuan Hu, Xin Jin, Mahmoud Khademi, Dongwoo Kim, Young Jin Kim, Gina Lee, Jinyu Li, Yunsheng Li, Chen Liang, Xihui Lin, Zeqi Lin, Mengchen Liu, Yang Liu, Gilsinia Lopez, Chong Luo, Piyush Madan, Vadim Mazalov, Arindam Mitra, Ali Mousavi, Anh Nguyen, Jing Pan, Daniel Perez-Becker, Jacob Platin, Thomas Portet, Kai Qiu, Bo Ren, Liliang Ren, Sambuddha Roy, Ning Shang, Yelong Shen, Saksham Singhal, Subhojit Som, Xia Song, Tetyana Sych, Praneetha Vaddamanu, Shuohang Wang, Yiming Wang, Zhenghao Wang, Haibin Wu, Haoran Xu, Weijian Xu, Yifan Yang, Ziyi Yang, Donghan Yu, Ishmam Zabir, Jianwen Zhang, Li Lyna Zhang, Yunan Zhang, and Xiren Zhou. 2025. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras. *CoRR*, abs/2503.01743.

Chenyang An, Zhibo Chen, Qihao Ye, Emily First, Letian Peng, Jiayun Zhang, Zihan Wang, Sorin Lerner, and Jingbo Shang. 2024. Learn from failure: Fine-tuning LLMs with trial-and-error data for intuitionistic propositional logic proving. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 776–790, Bangkok, Thailand. Association for Computational Linguistics.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *Preprint*, arXiv:2309.16609.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. 2024. A survey on evaluation of large language models. *ACM Trans. Intell. Syst. Technol.*, 15(3):39:1–39:45.

Zeming Chen, Gail Weiss, Eric Mitchell, Asli Celikyilmaz, and Antoine Bosselut. 2023. RECKONING: reasoning through dynamic knowledge encoding. 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*.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3882–3890. ijcai.org.

Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. 2021. Explaining answers with entailment trees. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7358–7370, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai

Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *Preprint*, arXiv:2501.12948.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. 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*.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Xiang Ren, Allyson Ettinger, Zaïd Harchaoui, and Yejin Choi. 2023. Faith and fate: Limits of transformers on compositionality. 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*.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. 2024. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7036–7050, Mexico City, Mexico. Association for Computational Linguistics.

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. *CoRR*, abs/2310.06825.

Dongwei Jiang, Marcio Fonseca, and Shay Cohen. 2024. LeanReasoner: Boosting complex logical reasoning with lean. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7497–7510, Mexico City, Mexico. Association for Computational Linguistics.

Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean-Bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Róbert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucinska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, and Ivan Nardini. 2025. Gemma 3 technical report. *CoRR*, abs/2503.19786.

Mehran Kazemi, Quan Yuan, Deepti Bhatia, Najoung Kim, Xin Xu, Vaiva Imbrasaite, and Deepak Ramachandran. 2023. 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*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Angeliki Lazaridou, Elena Gribovskaya, Wojciech Stokowiec, and Nikolai Grigorev. 2022. Internet-augmented language models through few-shot prompting for open-domain question answering. *CoRR*, abs/2203.05115.

Daniel Le Berre and Anne Parrain. 2010. The sat4j library, release 2.2, system description. *Journal on Satisfiability Boolean Modeling and Computation*, 7:59–64.

Seyed-Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2024. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *CoRR*, abs/2410.05229.

Yaniv Nikankin, Anja Reusch, Aaron Mueller, and Yonatan Belinkov. 2024. Arithmetic without algorithms: Language models solve math with a bag of heuristics. *CoRR*, abs/2410.21272.

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, Singapore. Association for Computational Linguistics.

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

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

Mohammed Saeed, Naser Ahmadi, Preslav Nakov, and Paolo Papotti. 2021. RuleBERT: Teaching soft rules to pre-trained language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1460–1476, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Abulhair Saparov and He He. 2023. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Kaya Stechly, Matthew Marquez, and Subbarao Kambhampati. 2023. Gpt-4 doesn't know it's wrong: An analysis of iterative prompting for reasoning problems. *Preprint*, arXiv:2310.12397.

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. 2024. Chain of thoughtlessness: An analysis of CoT in planning. *CoRR*, abs/2405.04776.

Alessandro Stolfo, Zhijing Jin, Kumar Shridhar, Bernhard Schoelkopf, and Mrinmaya Sachan. 2023. A causal framework to quantify the robustness of mathematical reasoning with language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 545–561, Toronto, Canada. Association for Computational Linguistics.

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, Online. Association for Computational Linguistics.

Jidong Tian, Yitian Li, Wenqing Chen, Liqiang Xiao, Hao He, and Yaohui Jin. 2021. Diagnosing the first-order logical reasoning ability through LogicNLI. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3738–3747, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Karthik Valmeekam, Kaya Stechly, and Subbarao Kambhampati. 2024. LLMs still can't plan; can LRMs? A preliminary evaluation of OpenAI's o1 on PlanBench. *CoRR*, abs/2409.13373.

Yuxuan Wan, Wenxuan Wang, Yiliu Yang, Youliang Yuan, Jen-tse Huang, Pinjia He, Wenxiang Jiao, and Michael Lyu. 2024. LogicAsker: Evaluating and improving the logical reasoning ability of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2124–2155, Miami, Florida, USA. Association for Computational Linguistics.

Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. 2023. Towards understanding chain-of-thought prompting: An empirical study of what matters. In *Proceedings of the 61st Annual Meeting of the Association for*

Computational Linguistics (Volume 1: Long Papers), pages 2717–2739, Toronto, Canada. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Zhe Yang, Yichang Zhang, Tianyu Liu, Jian Yang, Junyang Lin, Chang Zhou, and Zhifang Sui. 2024. Can large language models always solve easy problems if they can solve harder ones? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1531–1555, Miami, Florida, USA. Association for Computational Linguistics.

Honghua Zhang, Liunian Harold Li, Tao Meng, Kai-Wei Chang, and Guy Van den Broeck. 2023. On the paradox of learning to reason from data. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pages 3365–3373. ijcai.org.

A Sampling the Datasets

A.1 SAT Problem Generation

The algorithm to generate a dataset consisting of n problem instances, using the SAT sampling strategy, is described in Algorithm 1. For each problem instance, the algorithm first chooses the number of clauses m, by sampling this value uniformly at random between a lower bound cls_{min} and upper bound cls_{max}. Similarly, the total number ℓ of atomic propositions in the problem instance is chosen between at_{min} and at_{max}. Subsequently, the m premises are sampled by iteratively following the same process. First, the number of literals k is chosen to be 1 with probability of either 10% or 30%, depending on the specific scenario, while 2 is selected otherwise. Lower probability will induce problems with more inference depth. Then k atoms are randomly selected among the set of ℓ available atoms, and each atom is negated with 50% probability. Then, the algorithm checks whether the generated clause is consistent with the clauses that have already been added to the problem instance. If not, the newly generated clause is discarded. Finally, after the clauses have been generated, a literal is sampled to serve as the hypothesis of the problem instance, by choosing one of the ℓ atoms and negating it with 50% probability.

Algorithm 1 Generate SAT

GenSAT ($cls_{min}, cls_{max}, at_{min}, at_{max}, n$)
  for $idx = 1$ **to** n **do**
    $m =$ random(cls_{min}, cls_{max})
    $\ell =$ random(at_{min}, at_{max})
    // Generate the m premises
    while len($clauses$)$\leq m$ **do**
      $k =$ random(1, 2)
      $cls =$ GenerateClause (k, ℓ)
      $clauses \leftarrow$ Add(cls)
      if !isSatisfiable($clauses$) **then**
        $clauses \leftarrow$ Remove(cls)
      end if
    end while
    // Generate a hypothesis
    $hyp =$ GenerateClause (1, ℓ)
    if isEntailed($clauses, hyp$) **then**
      $label = True$
    else
      $label = False$
    end if
  end for
end

A.2 Sentence Templates

We translate the sampled clauses and literals into natural language sentences using a straightforward template-based approach. First, each atom is converted into a phrase like "Alice is great". We always use Alice as the subject of these phrases, but vary the adjective for different atoms. For our main experiments, the same adjectives are used for the problems in the training and test sets. However, in Appendix C.1, we will experiment with alternative test sets, in which a disjoint set of adjectives is used for the test set, and/or different names besides Alice are used as the subject. For the main experiments, we use a total of 500 distinct adjectives. To verbalize a negated atom, we put *not* in front of the adjective. For instance, if the atom a corresponds to the phrase "Alice is great", then $\neg a$ is verbalized as "Alice is not great". To verbalize clauses, we simply connect the verbalizations of the literals with the word *or*. For instance, if a corresponds to "Alice is great" and b corresponds to "Alice is timid" then $a \vee \neg b$ would be verbalized as "Alice is great or Alice is not timid".

A.3 RP and LP Sampling Strategies

The RP and LP strategies where introduced by Zhang et al. (2023). These strategies sample a

set of atoms, together with a set of Horn rules of the form $a_1 \wedge ... \wedge a_k \rightarrow b$. In accordance with our focus on simple reasoning problems, we limit their sampling process to only generate rules of the form $p \rightarrow q$, with a single atom in the body. Furthermore, to keep the format consistent with that of the SAT dataset, we convert each rule $p \rightarrow q$ into the logically equivalent clause $\neg p \vee q$, before translating them into sentences (in the same way as for SAT). For example, using this sampling[3], we get the following atoms *hesitant*, *spiky*, *curvy*, *hypercritical* and *bright*, as well as the following rules:

$$\textit{hesitant} \rightarrow \textit{spiky}$$
$$\textit{curvy} \rightarrow \textit{hypercritical}$$
$$\textit{curvy} \rightarrow \textit{bright}$$

together with the hypothesis *spiky* and the ground truth label *yes* (meaning that the hypothesis is entailed). We convert the rules into the following clauses:

$$\neg\textit{hesitant} \vee \textit{spiky}$$
$$\neg\textit{curvy} \vee \textit{hypercritical}$$
$$\neg\textit{curvy} \vee \textit{bright}$$

Finally, each atom (e.g. *hesitant*) is mapped to a phrase like "Alice is great", as for the SAT dataset.

A.4 Dataset Statistics

After generating the problem instances, we construct the training and test sets by selecting a balanced set of problem instances, ensuring there is an equal number of problem instances for each of the inference depths 0,1,2,3. Moreover, for each inference depth, we ensure that there is an equal number of positive and negative examples. An overview of the resulting datasets, along with some statistics, are shown in Table 2. Note that we always train the models on SAT.

B Experimental Settings

B.1 Model Checkpoints

Our experiments used DeepSeek-R1-Distill-Llama-8B (DeepSeek-AI et al., 2025), Meta-Llama-3-8B-Instruct (Touvron et al., 2023), Llama-3.1-8B-Instruct, Llama-3.2-3B-Instruct, Mistral-7B-Instruct (Jiang et al., 2023), Ministral-8B-Instruct-2410, Qwen2.5-7B-Instruct (Bai et al., 2023), Phi-4-mini-instruct (Abouelenin et al., 2025), and gemma-3-4b-it (Kamath et al., 2025). We fetched all pre-trained models from Huggingface platform using checkpoints listed in Table 3 and provided instructions both in training and testing. Note that we have focused on instruction-tuned LLMs, as such models perform better in straightforward and explicit tasks compared to the corresponding base models.

[3]We use the code available at https://github.com/joshuacnf/paradox-learning2reason

Statistics	**SAT**	**RP**	**LP**
Num. of premises	[2, 50]	[2, 50]	[2, 47]
Num. of atoms	[2, 15]	[2, 15]	[2, 15]
Depth	[0, 3]	[0, 3]	[0, 3]
Avg. literals	3.08	2.89	2.27
Training size	100K	-	-
Validation size	10K	-	-
Test size	1000	1000	1000

Table 2: The statistics of datasets in *default* settings.

LLM Checkpoints
deepseek-ai/DeepSeek-R1-Distill-Llama-8B
meta-llama/Meta-Llama-3-8B-Instruct
meta-llama/Llama-3.1-8B-Instruct
meta-llama/Llama-3.2-3B-Instruct
mistralai/Mistral-7B-Instruct-v0.3
mistralai/Ministral-8B-Instruct-2410
Qwen/Qwen2.5-7B-Instruct
microsoft/Phi-4-mini-instruct
google/gemma-3-4b-it

Table 3: LLM checkpoints used in the experiments.

B.2 Instruction Templates

The standard prompts that we used for our experiments are shown in Figure 6. We provided instructions that require the model to respond succinctly with "Yes" or "No". These full instructions were used during training, but we eliminated the last role during testing. Furthermore, we set the *temperature* to almost 0 in testing to ensure reproducibility with minimum variation.

For the (zero-shot) experiments with pre-trained GPT-4o and DeepSeek-R1-Distill-Llama-8B, we use prompt in Figure 7 and Figure 8. We set the *temperature* to 0.6 for DeepSeek-R1, according to the recommendation in the technical report. We experimented with several standard prompts for GPT-4o and chose the best-performing prompt for this comparison.

B.3 Fine-tuning Setup

We trained all models using QLoRA with 4-bit NF quantization and double quantization. Based on a recommendation in the QLoRA paper (Dettmers

```
Instruction for Llama family, Qwen, and Gemma

system: The input below provides a set of
premises and a hypothesis. Is the hypothesis
entailed by the premises? Answer with yes or no
only.
user: Premises: [insert premises]
Hypothesis: [insert hypothesis]
system: [Yes/No]

Instruction for Mistral family

user: The input below provides a set of
premises and a hypothesis. Is the hypothesis
entailed by the premises? Answer with yes or no
only. Premises: [insert premises]
Hypothesis: [insert hypothesis]
assistant: [Yes/No]

Instruction for Phi

system: The input below provides a set of
premises and a hypothesis. Is the hypothesis
entailed by the premises? Answer with yes or no
only.
user: Premises: [insert premises]
Hypothesis: [insert hypothesis]
assistant: [Yes/No]
```

Figure 6: Instruction templates that were used for fine-tuning.

et al., 2023), we applied LoRA to all linear layers of the models. For LoRA parameters, we set rank r at 64, α at 16, and drop out at 0.1. To fit the model in our GPU memory, we used batch size 4. We fine-tuned the models for 1 epoch. In addition, we used the AdamW optimizer with a learning rate set to 2×10^{-4}. All experiments were conducted on a single NVIDIA RTX 4080 SUPER. It took 20 to 37 GPU hours to fine-tune one language model on default SAT training and 4 minutes to test per case.

C Additional Experimental Results

Table 4 summarizes the performance of Llama-3.1-8B-instruct fine-tuned on SAT, with different numbers of duplicates added and for different reodering strategies. The performance gradually deteriorates when duplicate premises are introduced, especially for RP and LP, with the accuracy dropping up to 6% in both cases. Furthermore, the model is also

```
Standard prompt for GPT-4o and o3-mini

system: Do the premises entail the hypothesis?
Answer with yes or no only.
user: Premises: [insert premises]
Hypothesis: [insert hypothesis]

CoT prompt for GPT-4o and o3-mini

system: Carefully analyze the logical
structure of the following premises and
hypothesis. Break down your reasoning into
step-by-step derivations. Evaluate whether the
hypothesis is strictly entailed by the
premises. If the hypothesis follows logically,
return Yes. Otherwise, return No.

Format your response as follows:

Explanation: [Step-by-step derivation]
Entailment: [Yes/No]

user: Premises: [insert premises]
Hypothesis: [insert hypothesis]
```

Figure 7: Standard and CoT prompts for zero-shot results obtained with closed-source GPT-4o and o3-mini models.

sensitive to literal positioning, with drops up to 10%. This result is consistent with Section 4 where the *first* strategy leads the worst performance.

C.1 Sensitivity to Changes in Verbalization

We adjusted the default SAT test by changing the verbalization in three ways. In the *new subjects* scenario, we used 500 random subject names besides "Alice" (for the test set only). We ensure that all premises from the same problem instance always refer to the same subject. To analyze model sensitivity to predicate changes, we replaced the adjectives with 500 color names in the *new predicates* scenario. Lastly, in the *new templates* scenario, we describe the formulas as rules. For instance $a \vee b$ would be translated to $\neg a \rightarrow b$ and verbalized as "If [verbalization of $\neg a$] then [verbalization of b]". Table 6 shows an example of the verbalizations used in each of the considered variants. Table 5 summarizes the resulting model performance. The model is generally robust to changes in subjects and predicates, while struggling with the template change (especially for $k = 2$ and $k = 3$).

Test Sets	SAT				RP				LP			
	none	end	shuf	first	none	end	shuf	first	none	end	shuf	first
default	99.1	99.2	98.3	97.7	**99.0**	99.0	94.1	89.5	**99.1**	99.1	94.1	89.4
1_dupl	99.1	99.2	98.3	98.2	98.1	98.3	94.8	89.7	98.5	99.0	93.6	90.4
2_dupl	98.9	98.7	98.4	98.2	98.2	98.7	94.6	90.6	98.3	98.2	94.4	90.9
3_dupl	98.3	98.3	97.6	97.7	98.0	98.3	94.5	90.5	97.6	97.8	95.6	92.1
5_dupl	98.6	98.4	98.3	97.8	95.5	96.2	93.4	90.3	96.8	96.5	94.7	92.8
7_dupl	97.8	97.6	97.8	97.2	95.4	95.4	92.7	90.5	96.0	94.9	93.3	92.4
10_dupl	98.0	97.8	98.0	97.3	93.2	93.4	92.5	89.7	94.9	93.5	94.0	93.2

Table 4: Test accuracy across all sampling for fine-tuned Llama-3.1-8B-instruct on SAT. We varied the scenarios using order types and duplicated premises. Test sets are balanced with a random guess of 50%. The underlined result shows the in-distribution test, whereas the results in bold represent out-of-distribution tests. Note that some examples in LP *default* may contain duplicated premises.

Test Sets	SAT				RP				LP			
	k=0	k=1	k=2	k=3	k=0	k=1	k=2	k=3	k=0	k=1	k=2	k=3
default	100	99.2	99.6	97.6	100	99.6	99.2	97.2	100	99.6	98.8	98.0
new subject	100	98.8	100	96.8	100	99.6	98.8	97.2	100	99.6	98.4	98.0
new adjectives	100	98.8	99.2	96.4	100	100	98.8	95.6	100	100	98.8	97.6
new templates	99.6	93.6	86.0	76.0	100	90.8	74.8	66.0	100	84.0	66.4	60.8

Table 5: Test accuracy on different changes in verbalization.

Scenario	Clauses
default	Alice is noisy or Alice is cheerful
new subjects	Henry is noisy or Henry is cheerful
new predicates	Henry is blue or Henry is ivory
new templates	If Alice is not noisy then Alice is cheerful

Table 6: A clause transformation in verbalization tests.

C.2 o3-mini Analysis

Even though o3-mini deliberately performs step-by-step reasoning even with our standard prompt, we noticed that the model slightly improved if we explicitly instructed it to do so, especially for problems of higher depth. Apart from constraining the final answer to yes/no only, we let the model respond in a CoT manner and relax the maximum number of new tokens. Figure 7 shows the prompts used for o3-mini test.

In Table 1, we found that o3-mini's performance is very close to 100% across all depths with this CoT prompt. Moreover, the few mistakes that o3-mini makes can be explained by potentially misleading verbalizations in the problem instances, rather than genuine reasoning errors. For example, *podgy* and *overweight* are considered to have the same meaning by o3-mini, whereas in our problem instances they are supposed to be independent properties. Similarly, *lively* and *listless* are believed to be in contradiction by o3-mini.

```
CoT prompt for DeepSeek-R1

Below is an instruction that describes a task,
paired with an input that provides further
context. Write the response that appropriately
completes the request. Before answering, think
carefully about the question and create a
step-by-step chain-of-thoughts to ensure a
logical and accurate response.

### Instruction:
Evaluate whether the hypothesis is strictly
entailed by the premises. If the hypothesis
follows logically, return Yes. Otherwise,
return No.
Format your response as follows:
Entailment: [Yes/No]

### Question:
Premises: [insert premises]
Hypothesis: [insert hypothesis]

### Response:
<think>
{}
</think>
{}
```

Figure 8: CoT prompt for zero-shot test in DeepSeek-R1 model.