On Memorization of Large Language Models in Logical Reasoning

Chulin Xie[1,2], Yangsibo Huang[1,3], Chiyuan Zhang[1], Da Yu[1], Xinyun Chen[1], Bill Yuchen Lin[4], Bo Li[2], Badih Ghazi[1], Ravi Kumar[1]

[1]Google, [2]University of Illinois Urbana-Champaign,
[3]Princeton University, [4]University of Washington

Abstract

Large language models (LLMs) achieve good performance on challenging reasoning benchmarks, yet could also make basic reasoning mistakes. This contrasting behavior is puzzling when it comes to understanding the mechanisms behind LLMs' reasoning capabilities. One hypothesis is that the increasingly high and nearly saturated performance on common reasoning benchmarks could be due to the memorization of similar problems. In this paper, we systematically investigate this hypothesis with a quantitative measurement of memorization in reasoning tasks, using two dynamically generated logical reasoning benchmarks based on Knights and Knaves (K&K) puzzles and Zebra puzzles (DynamicZebra). We find that LLMs could interpolate and memorize the training puzzles (achieving near-perfect accuracy) after fine-tuning, yet they struggle with slight variations of these puzzles. On the other hand, we show that while fine-tuning leads to heavy memorization, it also consistently improves generalization performance. Through in-depth analyses with perturbation tests, cross difficulty-level transferability, probing model internals, and fine-tuning with wrong answers, we establish that LLMs develop reasoning skills on logical puzzles alongside memorization. Finally, our analysis based on a per-sample memorization score sheds light on how LLMs switch between reasoning and memorization when solving logical puzzles. Our code and data are available at `https://memkklogic.github.io/`.

1 Introduction

Modern Large Language Models (LLMs) show impressive reasoning capabilities that allow them to solve a wide range of challenging problems including commonsense reasoning and mathematical reasoning. In the meantime, LLMs also make mistakes on some of the most basic problems, such as comparing which number is bigger—13.11 or 13.8 (Lin, 2024), and counting the number of sisters that Alice's brother has (Nezhurina et al., 2024). This contrast is puzzling when it comes to understanding how exactly LLMs solve reasoning tasks. This question is important both scientifically and practically: understanding how LLMs reason could shed light on their learning and generalization behaviors. It is also crucial for real-world applications where robust reasoning is required due to safety and trustworthiness concerns (Wang et al., 2023a; Wallace et al., 2024; Lee et al., 2024).

One hypothesis is that LLMs could be relying on *memorization* when solving those reasoning tasks, especially when measured by popular benchmarks that could be accidentally leaked into various massive internet-crawled pre-training datasets. Previous work (Tirumala et al., 2022; Carlini et al., 2023) show that LLMs could indeed memorize the training data, which may lead to potential privacy (Carlini et al., 2021) or copyright (Karamolegkou et al., 2023; Wei et al., 2024) concerns. Additional evidences of potential memorization come from extensive studies on data contamination in LLMs (Balloccu et al., 2024; Xu et al., 2024). To mitigate the issue of benchmark saturation potentially due to memorization, some papers focus on designing dynamic benchmarks (Srivastava et al., 2024; Jain et al., 2024) or alternative evaluation protocols (Xu et al., 2024; Srivastava et al., 2024).

In this paper, we take a direct approach to quantify the memorization behaviors of LLMs in reasoning tasks within a controlled setting. Specifically, we seek to understand: (i) whether LLMs rely on memorization to solve reasoning tasks, and (ii) whether memorization is only detrimental to learning to reason. Both questions are inspired by human behavior. For instance, when a student works hard on the preparation material for an exam, the preparation could help them get familiarized with the problems, and their ability to solve new problems could usually improve with enough ex-

Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 2742–2785

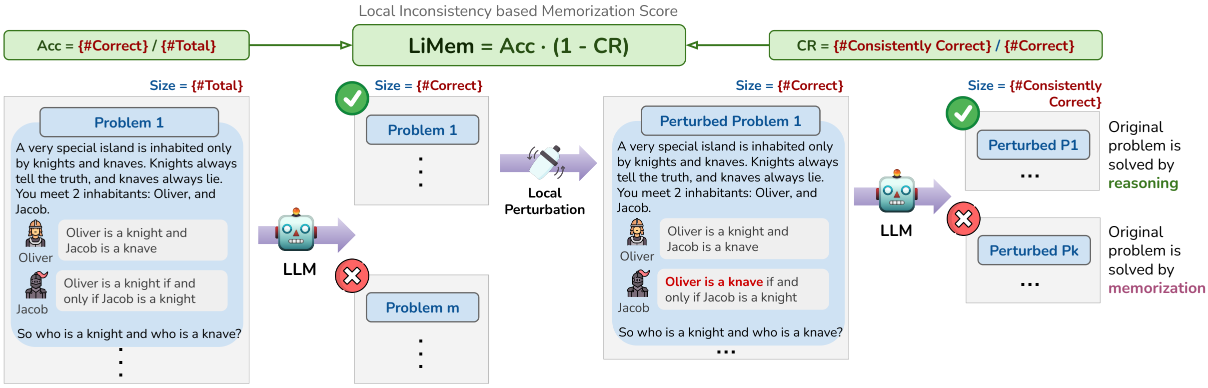


Figure 1: Illustration of the definition of Local Inconsistency based Memorization Score, LiMem. High level of memorization occurs when the model shows high accuracy in solving some problems but fails to consistently solve those problems under local perturbations that require similar underlying reasoning principles.

ercises. However, without genuinely understanding the principles, they might fail when the same problem is slightly changed despite doing well on prepared problems. Our metric of memorization LiMem (Fig. 1) is based on this intuition: an LLM shows a high level of memorization when it solves reasoning problems with high accuracy but struggles to consistently solve those problems under local perturbations requiring similar mathematical principles (i.e., low consistency). We note that a similar perturbation idea (but mostly at *language-level*) has been used in previous work, especially in detecting contamination (Golchin and Surdeanu, 2023; Xu et al., 2024). However, given our focus on understanding memorization in logical reasoning tasks, we further consider *problem-level* perturbation that slightly changes the mathematical structure of a puzzle, in addition to language-level perturbations. To facilitate our study, we propose two new logical reasoning benchmarks that support automatic problem-level perturbation. With these tools, we evaluate the reasoning power of 17 LLMs. We then fine-tune Llama3-8B and GPT4o-mini to quantify memorization in reasoning tasks, and reveal interesting interplay: while LLMs indeed tend to memorize many training logical puzzles, they also develop reasoning capabilities during fine-tuning (even directly on question-answer pairs without reasoning steps), and the reasoning performance improves when memorizing more training puzzles. We summarize key contributions:

- To quantify memorization in reasoning tasks, we define a memorization score based on the notions of performance inconsistency under local perturbation, inspired by human behavior (§ 2.1).
- To facilitate the measurement, we propose a new logical reasoning benchmark based on the *Knights and Knaves* (K&K, Smullyan, 1978; Johnson-Laird and Byrne, 1990) puzzles, that can generate new puzzles at different difficulty levels, locally perturb existing puzzles, and automatically synthesize detailed reasoning steps to solve a given puzzle (§ 2.2).
- We show that K&K puzzles are challenging, and only the most advanced LLMs could solve them well. Moreover, our analysis suggests those models exhibit some level of memorization (§ 3).
- By fine-tuning on K&K samples, we confirm that modern LLMs are capable of memorizing a large collection of puzzles, and reach high memorization score when interpolating (i.e., fitting, Belkin et al., 2018) the training set. We observe that the models' generalization accuracies continue to improve as memorization grows (§ 4).
- We design various in-depth analyses (§ 4.1∼§ 4.2) to verify that LLMs developed improved reasoning capabilities (i.e., generalization) after fine-tuning even with only question-answer pairs, via local perturbation tests, cross difficulty-level transferability, fine-tuning with wrong answers, and model internal probing.
- We show that fine-tuning with detailed reasoning steps can further boost the generalization on K&K puzzles, even when fine-tuned with wrong reasoning steps (§ 5).
- To analyze the interplay between memorization and reasoning, we measure per-sample memorization and study how LLMs switch between memorization and reasoning (§ 6).
- To verify the generalizability of our findings, we propose *DynamicZebra*, another dynamically generated benchmark based on a different family of logical puzzles (§ 2.3), and present similar

empirical results to K&K in § 7.

2 Measuring Memorization in Reasoning

2.1 Memorization Metrics for Reasoning

Memorization of LLMs has been studied in various contexts such as privacy (Carlini et al., 2023), copyright (Carlini et al., 2021), and solving knowledge intensive tasks (Hartmann et al., 2023). In this paper, we are specifically interested in measuring the level of memorization when solving reasoning tasks, by borrowing intuition from human behavior. For example, when preparing for an exam, a student may not be able to fully digest the underlying principles due to various reasons or constraints. But when (luckily) facing the same problem the student had prepared for, they would still be able to solve it. A key characteristic of this type of memorization is: (A) high accuracy on observed problems and (B) low accuracy when the problem is slightly changed. Based on this intuition, for a dataset $\mathcal{D}$ of reasoning puzzles, we combine the following two quantities to measure memorization:

1. For (A), we measure the accuracy of a target model f on $\mathcal{D}$, denoted as $\mathsf{Acc}(f;\mathcal{D})$. We are especially interested in measuring on the set of *observed puzzles*, i.e., the training set, $\mathsf{Acc}(f;\mathsf{Tr})$. We say f *interpolates* (Belkin et al., 2018; Belkin, 2021) the training puzzles if $\mathsf{Acc}(f;\mathsf{Tr}) \approx 100\%$.
2. For (B), we measure a *consistency ratio* $\mathsf{CR}(f;\mathcal{D})$ between the number of *consistently solved puzzles* after some *local perturbations*, and the number of originally solved puzzles. We are interested in local perturbations that make minimal changes to the puzzle and maintain the same underlying principle for solving it, and a similar difficulty level (to be specified in § 2.2).

We combine the two factors to define a *Local Inconsistency-based Memorization Score* $\mathsf{LiMem}(f;\mathcal{D}) \in [0,1]$:

$$\begin{aligned}\mathsf{LiMem}(f;\mathcal{D}) &= \mathsf{Acc}(f;\mathcal{D}) \cdot (1-\mathsf{CR}(f;\mathcal{D})) \qquad (1)\\ &= \frac{\texttt{\#Correct} - \texttt{\#Consistently\_Correct}}{\texttt{\#Total}}.\end{aligned}$$

When there is no ambiguity, we call it the memorization score. A larger score provides stronger evidence of memorization (i.e., a larger proportion of memorized examples in the given dataset). Specifically, a high $\mathsf{LiMem}(f;\mathsf{Tr})$ matches the characteristic behavior of human memorizing observed puzzles, and in this case we say f *memorized* the training puzzles. Note that the $\mathsf{Acc}(f;\mathcal{D})$ factor is necessary, as there can be three types of behaviors: (i) solving by memorization, (ii) solving by reasoning, (iii) not solving (e.g., random guessing). A high $\mathsf{LiMem}(f;\mathcal{D})$ indicates (i), but a low $\mathsf{LiMem}(f;\mathcal{D})$ would only indicate (ii) if we separately check that $\mathsf{Acc}(f;\mathcal{D})$ is high.

To effectively measure the memorization score $\mathsf{LiMem}(f;\mathcal{D})$, we need a principled way to (1) locally perturb the puzzle while maintaining its difficulty level; (2) compute the new correct answer after perturbation. Towards this goal, we design and implement a functional logical reasoning dataset based on the Knights and Knaves puzzles.

2.2 Knights and Knaves Benchmark

Knights and Knaves (K&K) is a type of logical puzzle, where the goal is to infer each character i's truthfulness B_i (Boolean value) by judging the logical consistency of the statements S_i they made. Fig. 2 shows an example.

The principle underlying K&K is the Boolean satisfiability problem (SAT) (Boolean satisfiability problem). SAT was the first problem proven to be NP-complete and many well-known problems can be translated into SAT, such as hardware and software verification and theorem proving (SAT solver, 2024). Hence, the performance of a model on SAT (i.e., K&K puzzles) can be important indicative of its reasoning capabilities. Specifically, consider a K&K puzzle involving N people, and a possible Boolean value assignments to $\{B_i\}_{i=1}^N$, where B_i indicates whether the ith person is telling the truth, i.e., their statement S_i is true. Therefore, a valid solution to a K&K puzzle is an assignment such that the following formula is true: $(B_1 \Leftrightarrow S_1) \wedge (B_2 \Leftrightarrow S_2) \wedge \cdots \wedge (B_N \Leftrightarrow S_N)$.

Based on the K&K puzzle, we design a *dynamic* benchmark that supports generating new puzzles and local perturbations. Our benchmark has 2 modules (See Fig. 2 for an overview):

The Abstract Module contains 4 components that generate and manipulate K&K puzzles in an abstract form: The *Generator* that produce random puzzles; the *Solver* that find valid solutions algorithmically; the *Reasoner* that generate human-like reasoning steps (chain-of-thoughts, CoT); and the *Perturber* that maps a given puzzle to a local perturbation. Each puzzle involves N people, each making a statement forming a logical tree with max width/depth of W/D, using the logical operations *and*, *or*, *not*, *implication*, and *equivalence*. The *Per-*

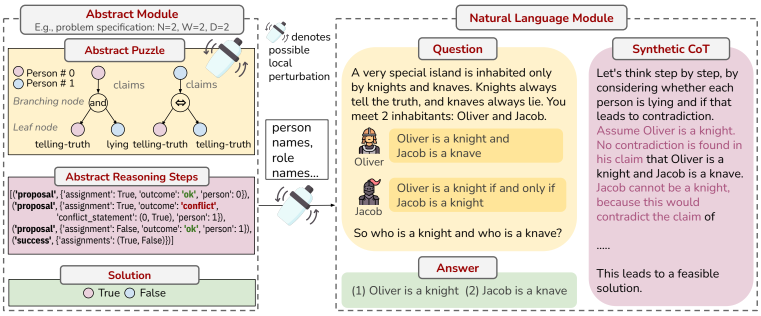


Figure 2: K&K data generation framework employs abstract module and natural language module to generate question-answer pair and synthetic CoTs for each K&K puzzle, based on the problem specification: number of persons (N), tree width (W), and depth (D). Perturbers in these modules can alter the math structure and language description, respectively, and recompute the QA pair.

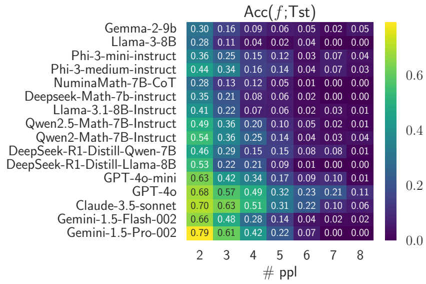


Figure 3: Test acc of off-the-shelf LLMs under 0-shot direct prompting drops with increasing puzzle complexity. For reference, OpenAI o1 with test-time compute achieves an acc of 0.86 (0.67) on 8-ppl (18-ppl) task.

turber is the most important part that is not usually supported in previous benchmarks. It generates mathematical perturbation by replacing a statement or a leaf node in a statement with a newly sampled one, and ensures the perturbation has a different solution. This support is crucial for making our memorization measurements.

The Natural Language Module converts the puzzles and the generated CoTs into natural language. It uses random names and templates to diversify the generated puzzles, and also supports *language level perturbation* to a given puzzle.

See § B for more details. We generate and release a core dataset of 1000/100/50 train/test/validation puzzles for each $2 \leq N \leq 8$ people[1]. By default we use a maximum tree width of $W = 2$ and depth $D = 2$. For each puzzle, we also generate six perturbed variants (2 problem level and 4 language level): {*perturbed statement, perturbed leaf node, random role-pair name, uncommon person name, reordered statement, flipped role*}.

K&K is challenging for off-the-shelf models. We use 0-shot direct prompting with task-specific instructions for open-ended question-answering (details in § D.2).[2] Accuracy is determined by keyword matching and requires correctly identifying *all* characters in the conclusion. We evaluate 17 leading models known for strong reasoning performance. Fig. 3 shows that K&K puzzles are highly challenging—even for the simplest 2-ppl puzzles, the best models (except o1 model) achieve at most 70% accuracy, which drops to just 11% for 8-ppl puzzles. In § E.1, we show that various prompting techniques like CoT/1-shot/self-consistency (Wang et al., 2023b) cannot fundamentally improve performance on challenging K&K tasks.

2.3 DynamicZebra

To demonstrate the generalizability of our dynamic data generation principles to other reasoning tasks, we introduce DynamicZebra, a novel dataset of fully controlled and perturbed *Zebra puzzles* (i.e., Einstein puzzle (Zebra Puzzle))[3]. DynamicZebra comprises: $5k$ training puzzles across 5 difficulty levels (4×4–5×5), each with two perturbed versions; Perturbations involve swapping values within specific attributes and updating the clues to maintain the original solution grid; Test puzzles spanning 12 difficulty levels (3×3–6×6, each with 100 puzzles). For more details, see § C.

3 Quantify Memorization in Reasoning

Here, we study a model's memorization behavior when fine-tuned on K&K puzzles.

Fine-tuning setup. We fine-tune the models for each N-people task separately, with $n_{\text{train}} = 1,000$ for $3 \leq N \leq 8$, and $n_{\text{train}} = 200$ for 2-people task due to limited number of combinations. We take Llama3-8B and GPT4o-mini and run *supervised fine-tuning* (SFT) on a set of K&K training puzzles disjoint from the test set. We consider two fine-tuning paradigms: (1) Fine-tuning on detailed CoT steps (**CoT FT**): during SFT, the model observes the concatenation of the question, synthetic CoT

[1]For 2-ppl we only include 200 training puzzles due to the limited problem space. Note the problem space is huge as N increases: e.g., for 8-ppl ($D, W = 2, 2$), there are ~10^{24} unique problems, and ~30% of them has a unique solution based on empirical estimation from 100k random generations.

[2]Even under direct prompting, capable LLMs can generate CoT. Our evaluation mainly considers the 0-shot setting to avoid biases from in-context examples (Zhao et al., 2021), but we provide results for CoT prompting, 1-shot prompting & self-consistency prompting in § E.

[3]While Lin et al. (2025) also present a Zebra puzzle dataset with $1k$ samples, their generation algorithm is unavailable and they do not offer perturbed puzzles.

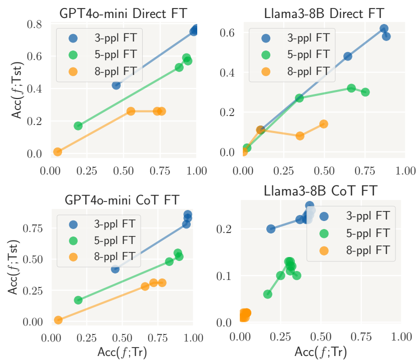


Figure 4: Train & test accuracy increases over the epochs. FTed LLMs can achieve interpolation ($\approx 100\%$ train accuracy) for easy tasks, e.g., 3/5-ppl puzzles. Llama3-8B struggles with CoT FT on K&K tasks, likely due to limited model capacity.

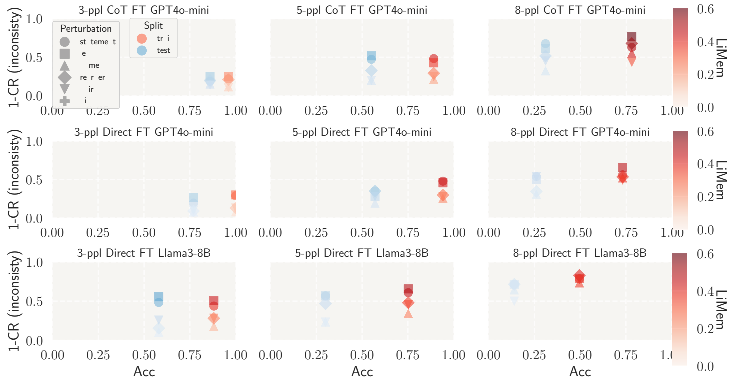


Figure 5: Fine-tuned LLMs generally exhibit both higher clean accuracy (x-axis) & inconsistency ratio under perturbations (y-axis) on the train set than test set, resulting in a higher memorization score (color spectrum). LLMs show stronger memorization under math-level perturbations (statement/leaf) than language level. We separately report memorization score in Fig. 21 and consistency ratio in Fig. 22, and results under combined math & language-level perturbations in Fig. 23.

steps, and the answer for each puzzle; the loss is computed on the CoT steps and the answer part. (2) Fine-tuning on the answers (**Direct FT**) where the model observes the question-answer pair for each puzzle, and the loss is only computed on the answer part. Examples of CoT FT/Direct FT training instances are provided in § D.2.2. We fine-tune Llama3-8B for 50 epochs[4] and GPT4o-mini for 5 epochs via the OpenAI fine-tune API (details in § D.2). During the evaluation, we follow the same prompting paradigm as FT paradigm, i.e., direct/CoT prompting for direct/CoT-FTed model, which is shown effective in § E.3.

LLMs interpolate K&K training puzzles. In Fig. 4, we present the training accuracy of models trained on each task on the x-axis (each dot represents a training epoch). We find that models exhibit high training accuracy in tasks such as $3/5$-people puzzles. The higher capacity model GPT4o-mini nearly achieves interpolation ($\mathsf{Acc}(f;\mathsf{Tr}) \approx 100\%$) using both Direct FT and CoT FT.

Interpolating LLMs have large memorization scores on training examples. From Fig. 5, **(1)** we observe high $\mathsf{LiMem}(f;\mathsf{Tr})$ memorization score on training samples (e.g., $\sim 50\%$ on 8-people task) under various perturbations. It shows significant gaps between accuracy on the original sample and the consistent accuracy under perturbation, suggesting a heavy reliance on memorization. **(2)** $\mathsf{LiMem}(f;\mathsf{Tr})$ is higher for more difficult tasks (e.g., 5/8-people), which could mirror human behavior, where memorization is often used to tackle challenging tasks that people do not fully understand. **(3)** The more capable LLM GPT4o-mini generally show lower memorization scores.

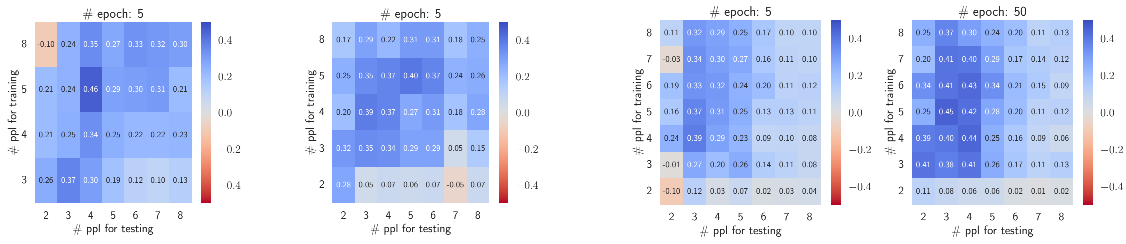
Ablation on local perturbations. Comparing different perturbations in Fig. 5, we find that **(1)** LLMs exhibit a higher memorization score when evaluated with math-level perturbations (e.g., statement/leaf) compared to language-level, which indicates that LLMs can compose the language understanding capability to solve the same puzzle in alternative phrasing. **(2)** LLMs get nearly zero accuracy on role-flipped samples (e.g., when a knight, typically viewed as truthful, is defined as always lying), and memorization score $\mathsf{LiMem}(f;\mathsf{Tr})$ under role-flipping for Llama3-8B is $\sim 80\%$ as shown in Fig. 7. This could be due to an internal bias or commonsense understanding that knights are inherently good characters (e.g., truthful), and thus LLMs disregard the altered puzzle statement.

4 Learn to Reason by Fine-tuning With Answers Only

§ 3 shows that fine-tuned models exhibit memorization when solving K&K reasoning tasks. Does it mean that those models do not have reasoning capabilities at all? Here we show that LLMs can do both, and the reasoning capability consistently improves as the memorization level increases when the models are fine-tuned on K&K puzzles.

We focus on analyzing Direct FT in this section and discuss CoT FT in § 5. For humans, solving K&K tasks without understanding the underlying logic is difficult. However, after observing the step-by-step reasoning steps, people can understand the procedure and solve the puzzles more easily. Similarly, compared to CoT FT, learning from only

[4] We fine-tune Llama3-8B for max 100 epochs in Fig. 24 and find that it typically converges at 50 epochs.



(a) GPT4o-mini CoT FT (b) GPT4o-mini Direct FT (c) Llama3-8B Direct FT

Figure 6: Test accuracy improvement on N-people problems for LLMs fine-tuned on M-people problems, compared to the unfine-tuned model, under 0-shot direct prompting. Most grid values are above 0, indicating transferability and enhanced reasoning abilities across unseen tasks. Results for more epochs are in § E.3.

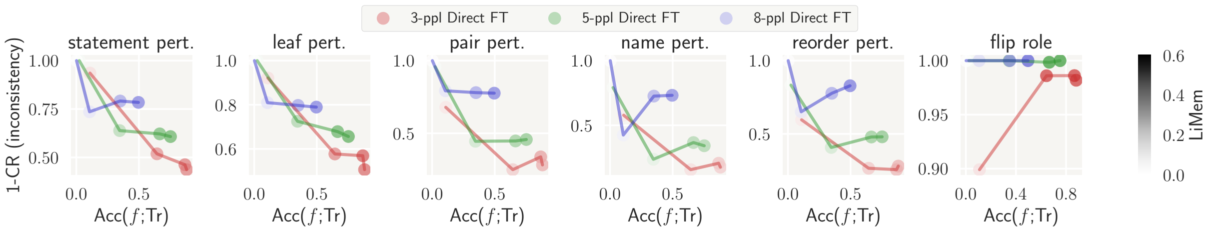


Figure 7: Inconsistency ratio (y-axis) on correctly solved training puzzles of fine-tuned Llama3-8B decreases over epochs (x-axis), even as the proportion of memorized training puzzles increases, as indicated by the larger $\mathsf{LiMem}(f;\mathsf{Tr})$ values (color).

answers (Direct FT) without detailed reasoning steps is intuitively more challenging for LLMs, as the models need to come up with the reasoning procedures on their own. Therefore, the models might be more likely to rely on memorization in this case. Surprisingly, from Fig. 5, we did not observe Direct FTed GPT4o-mini models exhibiting consistently higher memorization score than CoT FTed ones. It turns out that models can learn to reason K&K puzzles well directly from observing only question-answer pairs, as we will show in § 4.1. To better understand what the model learns through Direct FT, we conduct a probing analysis on model internals in § E.4 and an ablation study with incorrect answers fine-tuning in § E.4.

4.1 Reasoning Capabilities via Direct FT

Fine-tuned model generalizes across different difficulty levels. We evaluate LLMs' transferability by fine-tuning on M-people puzzles and testing on N-people puzzles. When $M \neq N$, the testing is out-of-distribution compared to training and solving it requires reasoning. The $N \times M$ test accuracy improvement grid (compared to the un-FTed model) in Fig. 6 shows: **(1)** Training on any M-people puzzle generally improves test accuracy on any N-people puzzles, suggesting that the model learns general task-solving rules after FT (to reason and solve both easier and harder unseen puzzles). **(2)** More training epochs (e.g., 50 vs. 5) improve generalization, especially for Llama3-8B. **(3)** Accuracy gains are larger for $N \leq 6$ puzzles, though improvements on harder tasks remain possible.

Inconsistency ratio decreases despite increased memorization. As shown in Fig. 7, the inconsistency ratio (y-axis) of fine-tuned LLMs on correctly solved training puzzles decreases over epochs, even as the memorization score $\mathsf{LiMem}(f;\mathsf{Tr})$ increases, indicating a higher proportion of memorized training puzzles (Eq. (1)). This reduction in inconsistency suggests a potential improvement in the model's generalization ability, aligning with its enhanced transferability observed in Fig. 6. The memorization score $\mathsf{LiMem}(f;\mathsf{Tr})$ under role-flipping is significantly higher than other perturbation, possibly due to an internal bias that knights are truthful. See Fig. 20 for results on GPT4o-mini.

Fine-tuning with 10k 8-people puzzles. Given the significant performance improvement from fine-tuning, a natural question arises: can brute-force fine-tuning on a very large number of puzzles eventually solve the K&K puzzles, by observing/memorizing a variety of combinations of persons' claims and their corresponding answers? We Direct FT GPT4o-mini on $1k$/$10k$ of the most challenging 8-people puzzles for 5 epochs. Fig. 9 shows that **(1)** $10k$-FT significantly outperforms $1k$-FT across all tasks, reaching $\sim 90\%$ test accuracy on moderately difficult 4/5-people puzzles. **(2)** CoT FT is generally more effective than Direct FT with $10k$ samples, likely due to the guidance provided by reasoning steps. We defer more discussion on Fig. 9 and results for Llama3-8B to § E.3.1.

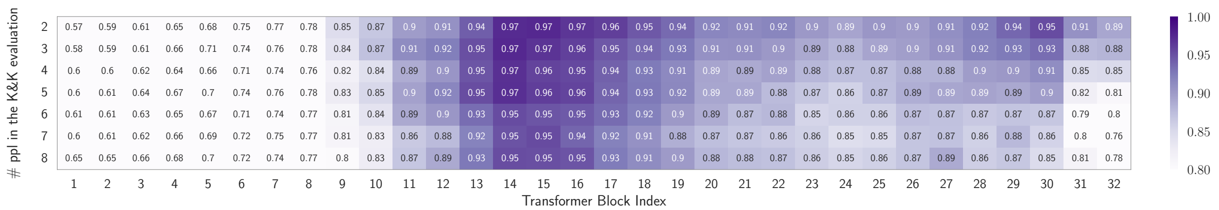


Figure 8: Probing accuracy of K&K puzzles with different number of people in testing puzzles across different layers of the Llama3-8B transformer model. Results for un-FTed models are shown in Fig. 38 in § E.

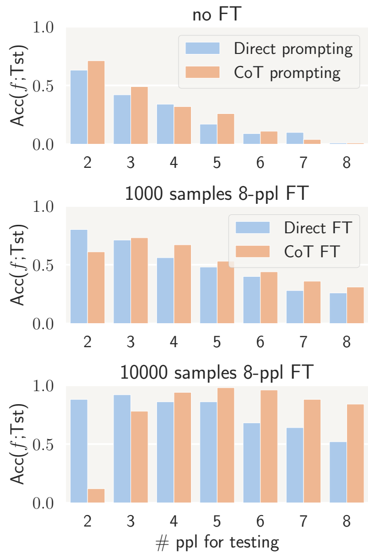


Figure 9: Transferability of $1k$/$10k$ 8-ppl FTed GPT4o-mini.

Additionally, we consider an extreme scenario of learning with incorrect answers. § E.4 shows that when training data includes $\leq 50\%$ incorrect answers, Direct FT still enhances K&K performance.

4.2 Probing Direct FTed Models

To investigate whether Direct FTed models develop internal understanding of the skills necessary to solve K&K puzzles when learning only from the answers, we use probing techniques (Adi et al., 2017; Conneau et al., 2018; Hewitt and Liang, 2019; Ye et al., 2024) to analyze their internal representations. Specifically, we study whether a Direct FTed model's intermediate outputs provide evidence that it can distinguish between correct and incorrect statements for a given K&K puzzle, which is essential for solving the puzzle via reasoning. For a given model, we extract intermediate outputs from all transformer blocks for 200 correct and 200 incorrect statements, then check whether these outputs form distinct clusters by measuring the training accuracy of a logistic regression model fit on them (see § D.2.3 for details). For each N-people K&K puzzle, we report the per-layer probing accuracy averaged across seven Direct FTed models, each FTed on an $M \in \{2, \ldots, 8\}$-people task. Fig. 8 shows **(1)** a clear trend of higher probing accuracy in deeper layers, peaking at around the 14th/15th layer. The near-perfect peak accuracy suggests that the model's internal representations have a clear distinction between true/false statements about a given puzzle. **(2)** The probing accuracy is much higher than the un-FTed model (Fig. 38 in § E), suggesting that such representations are learned from the question-answer pairs during Direct FT. **(3)** Puzzles with more people seem to demand more internal computation, as evidenced by the point where probing accuracy surpasses 85% shifting to later transformer blocks.

5 Learn to Reason by CoT Fine-tuning

Here we measure models' reasoning capabilities after fine-tuning with detailed reasoning steps.

Model learns to reason on CoT when model capacity is large enough. As shown in Fig. 4, **(1)** training with reasoning steps as guidance improves test accuracy (y-axis) on unseen puzzles. **(2)** However, Llama3-8B struggles with CoT FT, likely due to its limited capacity to effectively learn CoT skills with $\leq$1K training samples. **(3)** Similar to Direct FT results in § 4, in CoT FT, memorization of training data is higher than test data (Fig. 5), yet inconsistency ratio decreases despite that overall memorization score increases over training (Fig. 20), and the fine-tuned models show positive transferability to easier/harder tasks (Fig. 6). **(4)** Though LLMs can generalize surprisingly well under Direct FT, CoT FT could lead to much higher test accuracy, especially with a larger training set (Fig. 9).

Fine-tuning with wrong CoTs. To understand the role of the CoT component in improving model generalization during CoT FT, we fine-tune GPT4o-mini with two types of incorrect CoT data: **(a)** randomly shuffled CoT steps, disrupting the logic of the reasoning steps; and **(b)** CoTs with a single incorrect step, simulating genuine mistakes that people would sometimes make. The results in Fig. 10 show that **(1)** fine-tuning with a 100% corrupted CoT dataset can still enhance test accuracy over the epochs, suggesting that the model learns to rea-

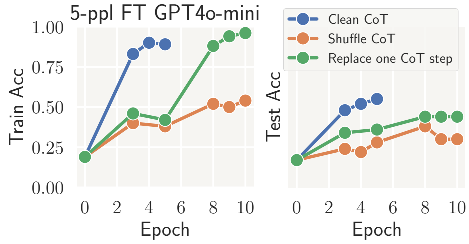


Figure 10: Fine-tuning GPT4o-mini with wrong CoTs.

son (potentially from the correct answers) despite CoT errors. **(2)** Altering one CoT step slows convergence and reduces test accuracy compared to clean CoT. **(3)** Shuffling CoT steps further harms both convergence and generalization. These also suggest that using correct logical chains in CoT can help LLMs to more effectively learn to reason.

6 Distinguish Memorizing vs. Reasoning

The findings above show that models' reasoning capabilities continue to improve as they memorize more training examples. In other words, the models use both memorization and reasoning to solve the puzzles. But how do they decide which examples to memorize or reason about? We can explore this by extending our memorization score to a per-example metric. Specifically, consider measuring Eq. (1) on a 1-point dataset $\mathcal{D} = \{x\}$. We skip the examples where $\mathsf{Acc}(f;\{x\}) = 0$ as the consistency ratio $\mathsf{CR}(\{x\})$ is NaN in this case. Here, $\mathsf{LiMem}(f;\{x\}) \in 0, 1$ acts as a binary indicator: 0 means x remains solvable after local perturbation, while 1 means it does not. Our goal is to determine if a clear rule separates these two types of puzzles.

We investigate the signatures of memorization for each model, based on data or model features. § E.6 show that data features (e.g., TF-IDF, Bag-of-Words, Word Length) can be informative and model features (e.g., embeddings) from the FTed model are consistently more informative than the un-FTed one—suggesting that the model's decision regarding memorization vs. reasoning on specific samples likely also stems from the FT process.

7 Results on DynamicZebra

To complement our findings on K&K, we validate the results on DynamicZebra, which is based on a completely different type of logical problem. The results in § E.7 show that while models rely on memorization to solve many training zebra puzzles and struggle to re-reason under small perturbations (Fig. 11), they also exhibit generalization to unseen test samples (Fig. 44).

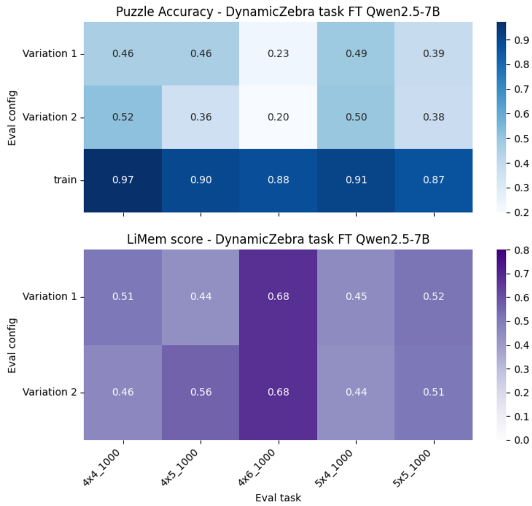


Figure 11: DynamicZebra training accuracy and memorization score on perturbed training set for Direct-FTed Qwen2.5-7B.

8 Related Work

Memorization in LLMs. Previous work on LLM memorization primarily focused on near-verbatim training text regurgitation from the perspective of privacy or copyright concerns (Carlini et al., 2021; Lee et al., 2022; Carlini et al., 2023; Lukas et al., 2023; Biderman et al., 2024; Prashanth et al., 2024). In contrast, we focus on quantifying the memorization behavior of LLMs when solving reasoning tasks, using a metric computed with the help of local perturbation of reasoning puzzles.

Benchmark Contamination and Logical Reasoning Evaluation. LLMs show performance drops on altered versions of popular reasoning benchmarks (Oren et al., 2024; Xu et al., 2024; Yang et al., 2023; Yao et al., 2024), indicating potential contamination. Synthetic benchmarks offer dynamic, scalable evaluation of logical reasoning (Giadikiaroglou et al., 2024; Parmar et al., 2024; Kazemi et al., 2024; Mondorf and Plank, 2024). Our work introduces comprehensive dynamic K&K and DynamicZebra puzzle sets with automatic generation of perturbations, solutions, and reasoning steps. Moreover, we define and measure memorization, and reveal its intricate relation to reasoning. We refer the readers to § A for a more comprehensive discussion of related work.

9 Conclusion

We propose a memorization metric LiMem based on the inconsistency when solving a locally perturbed logical reasoning puzzle, and quantitatively characterize the amount of memorization and reasoning with the proposed feature-rich dynamic logical reasoning benchmark.

Limitations

Our results reveal intricate phenomena of the interplay between reasoning and memorization, but challenging questions remain open: (i) While a model's reasoning capabilities improve during fine-tuning as it memorizes more training puzzles, it is unclear exactly how those capabilities develop, especially when fine-tuned on only question-answer pairs without detailed reasoning steps. (ii) Since some model-based indicators can approximately predict when the model is solving a specific puzzle by memorization vs by reasoning, can we further design intervention mechanisms to bias the model towards reasoning during inference or training time? Exploring the open questions in further research would deepen our understanding of this space.

Acknowledgments

We thank Yuntian Deng, Mingyang Deng, Ziqi Wang, Tiancheng Yu, Mike Mozer, Rishabh Agarwal, Danqi Chen, Matthew Jagielski, Nikunj Saunshi, Wei Xiong and Minghao Chen for their valuable feedback and discussions. YH acknowledges the support of the Wallace Memorial Fellowship and the compute resources at Princeton Language and Intelligence. BL acknowledges the support of NSF No. 2046726, NSF AI Institute ACTION No. IIS-2229876 and the Alfred P. Sloan Fellowship.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *ICLR*.

Simone Balloccu, Patrícia Schmidtová, Mateusz Lango, and Ondřej Dušek. 2024. Leak, cheat, repeat: Data contamination and evaluation malpractices in closed-source LLMs. *arXiv:2402.03927*.

Mikhail Belkin. 2021. Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation. *Acta Numerica*.

Mikhail Belkin, Daniel J Hsu, and Partha Mitra. 2018. Overfitting or perfect fitting? Risk bounds for classification and regression rules that interpolate. In *NeurIPS*.

Stella Biderman, Usvsn Prashanth, Lintang Sutawika, Hailey Schoelkopf, Quentin Anthony, Shivanshu Purohit, and Edward Raff. 2024. Emergent and predictable memorization in large language models. *NeurIPS*, 36.

Boolean satisfiability problem. Boolean satisfiability problem — Wikipedia, the free encyclopedia. [Online; accessed 20-Nov-2024].

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2023. Quantifying memorization across neural language models. In *ICLR*.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, and 1 others. 2021. Extracting training data from large language models. In *USENIX Security*.

Xinyun Chen, Ryan Andrew Chi, Xuezhi Wang, and Denny Zhou. 2024. Premise order matters in reasoning with large language models. In *ICML*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, and 1 others. 2024. Scaling instruction-finetuned language models. *JMLR*.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. In *IJCAI*.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. In *ACL*.

Yuntian Deng, Yejin Choi, and Stuart Shieber. 2024. From explicit cot to implicit cot: Learning to internalize cot step by step. *arXiv:2405.14838*.

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. 2023. Implicit chain of thought reasoning via knowledge distillation. *arXiv:2311.01460*.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jian, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Sean Welleck, Xiang Ren, Allyson Ettinger, Zaïd Harchaoui, and Yejin Choi. 2024. Faith and fate: Limits of transformers on compositionality. *NeurIPS*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, and 1 others. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv:2101.00027*.

Panagiotis Giadikiaroglou, Maria Lymperaiou, Giorgos Filandrianos, and Giorgos Stamou. 2024. Puzzle solving using reasoning of large language models: A survey. In *IJCAI*.

Shahriar Golchin and Mihai Surdeanu. 2023. Data contamination quiz: A tool to detect and estimate contamination in large language models. *arXiv:2311.06233*.

Vipul Gupta, David Pantoja, Candace Ross, Adina Williams, and Megan Ung. 2024. Changing answer order can decrease MMLU accuracy. *arXiv:2406.19470*.

Valentin Hartmann, Anshuman Suri, Vincent Bindschaedler, David Evans, Shruti Tople, and Robert West. 2023. SoK: memorization in general-purpose large language models. *arXiv:2310.18362*.

Luxi He, Yangsibo Huang, Weijia Shi, Tinghao Xie, Haotian Liu, Yue Wang, Luke Zettlemoyer, Chiyuan Zhang, Danqi Chen, and Peter Henderson. 2024. Fantastic copyrighted beasts and how (not) to generate them. *arXiv:2406.14526*.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *EMNLP*.

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2023. Large language models are reasoning teachers. In *ACL*.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *ACL*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. *arXiv:2403.07974*.

Bowen Jiang, Yangxinyu Xie, Zhuoqun Hao, Xiaomeng Wang, Tanwi Mallick, Weijie J Su, Camillo J Taylor, and Dan Roth. 2024. A peek into token bias: Large language models are not yet genuine reasoners. *EMNLP*.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? natural language attack on text classification and entailment. *AAAI*.

Philip N Johnson-Laird and Ruth MJ Byrne. 1990. Meta-logical problems: Knights, knaves, and rips. *Cognition*.

Antonia Karamolegkou, Jiaang Li, Li Zhou, and Anders Søgaard. 2023. Copyright violations and large language models. In *EMNLP*.

Mehran Kazemi, Quan Yuan, Deepti Bhatia, Najoung Kim, Xin Xu, Vaiva Imbrasaite, and Deepak Ramachandran. 2024. BoardgameQA: A dataset for natural language reasoning with contradictory information. In *NeurIPS*.

Seungone Kim, Se June Joo, Doyoung Kim, Joel Jang, Seonghyeon Ye, Jamin Shin, and Minjoon Seo. 2023. The cot collection: Improving zero-shot and few-shot learning of language models via chain-of-thought fine-tuning. In *EMNLP*.

Andrew Lee, Xiaoyan Bai, Itamar Pres, Martin Wattenberg, Jonathan K Kummerfeld, and Rada Mihalcea. 2024. A mechanistic understanding of alignment algorithms: A case study on DPO and toxicity. In *ICML*.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. Deduplicating training data makes language models better. In *ACL*.

Bill Yuchen Lin. 2024. Math Olympiad becomes easier for AI; Common sense is still hard.

Bill Yuchen Lin, Ronan Le Bras, and Yejin Choi. 2024. ZebraLogic: benchmarking the logical reasoning ability of language models.

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

Ziming Liu, Ouail Kitouni, Niklas S Nolte, Eric Michaud, Max Tegmark, and Mike Williams. 2022a. Towards understanding grokking: An effective theory of representation learning. *NeurIPS*.

Ziming Liu, Eric J Michaud, and Max Tegmark. 2022b. Omnigrok: Grokking beyond algorithmic data. In *ICLR*.

Nils Lukas, Ahmed Salem, Robert Sim, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Béguelin. 2023. Analyzing leakage of personally identifiable information in language models. In *S & P*.

R Thomas McCoy, Shunyu Yao, Dan Friedman, Mathew D Hardy, and Thomas L Griffiths. 2024. Embers of autoregression show how large language models are shaped by the problem they are trained to solve. *PNAS*, 121(41):e2322420121.

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. *arXiv:2410.05229*.

Philipp Mondorf and Barbara Plank. 2024. Liar, liar, logical mire: A benchmark for suppositional reasoning in large language models. *arXiv:2406.12546*.

Shikhar Murty, Pratyusha Sharma, Jacob Andreas, and Christopher D Manning. 2023. Grokking of hierarchical structure in vanilla transformers. In *ACL*.

Marianna Nezhurina, Lucia Cipolina-Kun, Mehdi Cherti, and Jenia Jitsev. 2024. Alice in wonderland: Simple tasks showing complete reasoning breakdown in state-of-the-art large language models. *arXiv:2406.02061*.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *ACL*, pages 4885–4901.

Yonatan Oren, Nicole Meister, Niladri Chatterji, Faisal Ladhak, and Tatsunori B Hashimoto. 2024. Proving test set contamination in black box language models. *ICLR*.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. 2023. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. In *EMNLP*.

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 *ACL*.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. 2022. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv:2201.02177*.

Akshara Prabhakar, Thomas L Griffiths, and R Thomas McCoy. 2024. Deciphering the factors influencing the efficacy of chain-of-thought: Probability, memorization, and noisy reasoning. *arXiv:2407.01687*.

USVSN Sai Prashanth, Alvin Deng, Kyle O'Brien, Jyothir SV, Mohammad Aflah Khan, Jaydeep Borkar, Christopher A Choquette-Choo, Jacob Ray Fuehne, Stella Biderman, Tracy Ke, and 1 others. 2024. Recite, reconstruct, recollect: Memorization in LMs as a multifaceted phenomenon. *arXiv:2406.17746*.

Haritz Puerto, Tilek Chubakov, Xiaodan Zhu, Harish Tayyar Madabushi, and Iryna Gurevych. 2024. Fine-tuning with divergent chains of thought boosts reasoning through self-correction in language models. *arXiv:2407.03181*.

Yasaman Razeghi, Robert L Logan IV, Matt Gardner, and Sameer Singh. 2022. Impact of pretraining term frequencies on few-shot numerical reasoning. In *Findings of EMNLP 2022*, pages 840–854.

Manley Roberts, Himanshu Thakur, Christine Herlihy, Colin White, and Samuel Dooley. 2023. To the cutoff... and beyond? A longitudinal perspective on LLM data contamination. In *ICLR*.

Joshua Robinson and David Wingate. 2023. Leveraging large language models for multiple choice question answering. In *ICLR*.

Abulhair Saparov and He He. 2023. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *ICLR*.

SAT solver. 2024. Sat solver — Wikipedia, the free encyclopedia. [Online; accessed 20-Nov-2024].

Raymond Smullyan. 1978. *What is the Name of this Book?* Prentice-Hall.

Saurabh Srivastava, Anto PV, Shashank Menon, Ajay Sukumar, Alan Philipose, Stevin Prince, Sooraj Thomas, and 1 others. 2024. Functional benchmarks for robust evaluation of reasoning performance, and the reasoning gap. *arXiv:2402.19450*.

Kushal Tirumala, Aram Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. 2022. Memorization without overfitting: Analyzing the training dynamics of large language models. *NeurIPS*.

Eric Wallace, Kai Xiao, Reimar Leike, Lilian Weng, Johannes Heidecke, and Alex Beutel. 2024. The instruction hierarchy: Training LLMs to prioritize privileged instructions. *arXiv:2404.13208*.

Boshi Wang, Xiang Yue, Yu Su, and Huan Sun. 2024a. Grokking of implicit reasoning in transformers: A mechanistic journey to the edge of generalization. In *NeurIPS*.

Boxin Wang, Weixin Chen, Hengzhi Pei, Chulin Xie, Mintong Kang, Chenhui Zhang, Chejian Xu, Zidi Xiong, Ritik Dutta, Rylan Schaeffer, and 1 others. 2023a. DecodingTrust: a comprehensive assessment of trustworthiness in GPT models. In *NeurIPS*.

Boxin Wang, Chejian Xu, Shuohang Wang, Zhe Gan, Yu Cheng, Jianfeng Gao, Ahmed Hassan Awadallah, and Bo Li. 2021. Adversarial GLUE: A multitask benchmark for robustness evaluation of language models. In *NeurIPS Datasets and Benchmarks*.

Haochun Wang, Sendong Zhao, Zewen Qiang, Bing Qin, and Ting Liu. 2024b. Beyond the answers: Reviewing the rationality of multiple choice question answering for the evaluation of large language models. *arXiv:2402.01349*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *ICLR*.

Boyi Wei, Weijia Shi, Yangsibo Huang, Noah A Smith, Chiyuan Zhang, Luke Zettlemoyer, Kai Li, and Peter Henderson. 2024. Evaluating copyright takedown methods for language models. In *NeurIPS Datasets and Benchmark*.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large language models are better reasoners with self-verification. In *EMNLP*.

Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. 2024. Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks. In *MAACL*, pages 1819–1862.

Ruijie Xu, Zengzhi Wang, Run-Ze Fan, and Pengfei Liu. 2024. Benchmarking benchmark leakage in large language models. *arXiv:2404.18824*.

Shuo Yang, Wei-Lin Chiang, Lianmin Zheng, Joseph E Gonzalez, and Ion Stoica. 2023. Rethinking benchmark and contamination for language models with rephrased samples. *arXiv:2311.04850*.

Feng Yao, Yufan Zhuang, Zihao Sun, Sunan Xu, Animesh Kumar, and Jingbo Shang. 2024. Data contamination can cross language barriers. *arXiv:2406.13236*.

Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. 2024. Physics of language models: Part 2.1, grade-school math and the hidden reasoning process. *arXiv:2407.20311*.

Zebra Puzzle. Zebra puzzle — Wikipedia, the free encyclopedia. [Online; accessed 18-May-2025].

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *NeurIPS*, 35:15476–15488.

Hugh Zhang, Jeff Da, Dean Lee, Vaughn Robinson, Catherine Wu, Will Song, Tiffany Zhao, Pranav Raja, Dylan Slack, Qin Lyu, and 1 others. 2024. A careful examination of large language model performance on grade school arithmetic. *arXiv:2405.00332*.

Tony Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *ICML*.

Wenjie Zhou, Qiang Wang, Mingzhou Xu, Ming Chen, and Xiangyu Duan. 2024. Revisiting the self-consistency challenges in multi-choice question formats for large language model evaluation. In *LREC-COLING*.

Kaijie Zhu, Jiaao Chen, Jindong Wang, Neil Zhenqiang Gong, Diyi Yang, and Xing Xie. 2024. Dyval: Graph-informed dynamic evaluation of large language models. In *ICLR*.

Yongshuo Zong, Tingyang Yu, Bingchen Zhao, Ruchika Chavhan, and Timothy Hospedales. 2024. Fool your (vision and) language model with embarrassingly simple permutations. *ICML*.

APPENDIX

A Extended Related Work

Memorization in LLMs. Prior work has explored training data memorization in LLMs, primarily in the contexts of *privacy* and *copyright* concerns (Carlini et al., 2021; Lukas et al., 2023; He et al., 2024), focusing on how LLMs may reproduce text near-verbatim to their training data (Lee et al., 2022; Carlini et al., 2023; Biderman et al., 2024). Prashanth et al. (2024) further introduces a taxonomy for memorization, categorizing it into Recitation, Reconstruction, and Recollection. They investigate the memorization behaviors of the Pythia model (Biderman et al., 2024) on the Pile dataset (Gao et al., 2020). In contrast, we examine memorization in the *reasoning* context, and focus on analyzing whether LLMs can accurately solve problems encountered during training but struggle to solve slightly perturbed variants. This allows us to better investigate the extent to which LLMs truly understand and generalize the *underlying principles* of the reasoning problems they have been trained on, as opposed to merely memorizing the *text*.

Recent research discusses signs of LLMs memorization in reasoning tasks by evaluating them on counterfactual reasoning tasks. These counterfactual tasks demand similar abstract reasoning skills as the original tasks but are less common in the training data. For instance, tasks such as reversing a sequence of words (McCoy et al., 2024) show better performance on high-probability sequences than on low-probability sequences; shifting each letter by n places in the alphabet (Rot-n) (Prabhakar et al., 2024; McCoy et al., 2024) demonstrates higher performance when $n = 13$ than for other values, likely because "Rot-13" is commonly used in online forums. (Wu et al., 2024) presents 11 counterfactual tasks (e.g., 1-indexing in Python, base-9 arithmetic) that show significant performance declines. (Jiang et al., 2024) changes some tokens in the reasoning task descriptions which leads to significant performance drops, suggesting that models might depend on recognizing superficial patterns with strong token bias. Moreover, (Razeghi et al., 2022) finds a strong correlation between the accuracy for a number on numerical reasoning tasks and its frequency in pretraining for GPT-J/GPT-Neo. In our study, we formally define a memorization score to quantify performance variance under task perturbations, covering both counterfactual alterations (e.g., switching the roles of knights and knaves) and standard perturbations on language level and problem structure level.

Detecting benchmark contamination. Recent work has shown that LLMs' performance drastically declines when faced with altered versions of popular reasoning benchmarks, suggesting potential contamination/memorization of these benchmarks. The benchmark variants include diverse forms such as altered multiple-choice questions formats (Wang et al., 2024b; Zong et al., 2024; Gupta et al., 2024; Zhou et al., 2024; Robinson and Wingate, 2023), rephrased or translated problems (Xu et al., 2024; Yang et al., 2023; Yao et al., 2024), shuffled example orderings (Oren et al., 2024), human-curated problems of comparable difficulty (Zhang et al., 2024), functional variants generating random instantiations (Srivastava et al., 2024; Mirzadeh et al., 2024), and problems beyond specific date cutoffs (Roberts et al., 2023; Jain et al., 2024). Previous work either focus on surface level language perturbations or require extensive expert-level annotations for math level variations. In contrast, our benchmark support automatic problem-level perturbation, solution and reasoning procedure generation, and easily scale to different difficult levels and dataset sizes without extra human efforts.

Logical reasoning benchmarks. To evaluate logical reasoning capabilities in LLMs, synthetic benchmarks have been developed to enable scalable generation of samples with varying configurations and difficulty levels (Clark et al., 2020; Giadikiaroglou et al., 2024; Parmar et al., 2024). For instance, DyVal (Zhu et al., 2024) uses directed acyclic graphs to dynamically generate samples on reasoning tasks including deductive, Boolean, and abductive reasoning. (Chen et al., 2024) focus on propositional logical problems involving definite clauses, and synthetically generate variations with different premise orders, such as forward, backward, and shuffled. (Dziri et al., 2024) explore the limitations of LLMs in tasks requiring compositional reasoning, including multiplication, logic grid puzzles, and dynamic programming problems. ZebraLogic (Lin et al., 2024) is an extended benchmark that systematically tests logical reasoning capabilities. BoardgameQA (Kazemi et al., 2024) presents a question-answering dataset characterized by contradictory facts and rules in the questions. PRONTOQA (Saparov and He, 2023) is a synthetic question-answering dataset where each example is generated from a synthetic world model represented in first-order logic. This dataset enables parsing the generated chain of thought into symbolic proofs, facilitating formal analysis. TruthQuest (Mondorf and Plank, 2024) is the most similar task to our work, which provides evaluation samples based on K&K-type of puzzles involving 3-6 person. Our work provides more comprehensive *dynamic* set of K&K puzzles that support automatic generation of perturbations, solutions and detailed reasoning steps. Moreover, based on this benchmark, we define and measure memorization in reasoning tasks, revealing intricate interplay between memorization and reasoning in LLMs.

Improving reasoning via fine-tuning. Prior work has explored fine-tuning LLMs on synthetic reasoning data to enhance their performance on reasoning. DyVal (Zhu et al., 2024) shows that fine-tuning Llama2-13B-chat on their synthetic reasoning benchmark improves its performance on other popular reasoning benchmarks. BoardgameQA (Kazemi et al., 2024) find that fine-tuning BERT-large and T5-XXL on their training dataset with synthetic proofs outperforms few-shot CoT prompting using PaLM. (Ye et al., 2024) pretrain GPT2 from scratch on synthetic math problems, synthetic CoT steps and solutions and show that model can solve problems from the same distribution and generalize to out-of-distribution (OOD) problems. However, (Dziri et al., 2024) show that while GPT-3 fine-tuned on their compositional reasoning tasks with/without reasoning steps can solve in-distribution (ID) problems, it fails to generalize to OOD tasks with increased problem sizes. Besides using synthetic CoTs, there are work using model-generated CoTs to enhance the models' reasoning capabilities (Chung et al., 2024). STaR (Zelikman et al., 2022) uses model self-generated CoTs on correctly solved samples to iteratively fine-tune itself as a self-taught reasoner. A number of work (Puerto et al., 2024; Kim et al., 2023; Ho et al., 2023; Hsieh et al., 2023) leverage CoTs generated from teacher models to train smaller student models. Additionally, some recent efforts have focused on leveraging intermediate reasoning steps in CoT more implicitly. For instance, Deng et al. (2023) distill intermediate reasoning tokens into the network layers by representing reasoning steps as vectors and using them as targets; Deng et al. (2024) distill CoT by gradually removing the intermediate steps and fine-tuning the model to internalize these steps, predicting the answers based on partial CoT. Both studies show that full CoT fine-tuning may not be necessary for the model to achieve strong reasoning performance.

In our study, we employ both direct fine-tuning and CoT fine-tuning to achieve memorization on K&K training data. Notably, our findings show that the fine-tuned GPT4o-mini and Llama3-8B models can effectively generalize to unseen OOD and ID K&K problems, contributing new insights to the topic of LLM fine-tuning for reasoning.

Orthogonal to our work, inference-time techniques have been explored to enhance reasoning performance such as self-consistency (Wang et al., 2023b), self-verification (Weng et al., 2023), and integration with external symbolic solvers (Pan et al., 2023).

Grokking. Our findings are related to Grokking, first identified by (Power et al., 2022) on a small algorithmic dataset, where validation accuracy suddenly improves from random chance to near-perfect generalization long after severe overfitting. Follow-up studies expanded the range of tasks where grokking occurs and proposed various explanations (Liu et al., 2022a; Murty et al., 2023; Liu et al., 2022b). Recently, (Wang et al., 2024a) observed grokking in the domain of complex knowledge-based tasks, showing that implicit reasoning over parametric knowledge emerges only after extensive overfitting. In this work, we observe a related phenomenon but through the lens of memorization and logical reasoning. Through novel (math & language-level) perturbation tests and transferability analyses, we verify that LLM's reasoning skills emerge alongside memorization. Furthermore, our investigation focuses on logical reasoning, offering new insights into how LLMs acquire logical reasoning skills.

Adversarial robustness under perturbations. Language-level perturbations have been widely used to assess the adversarial robustness of language models, often involving manually annotated attacks, as seen in advGLUE (Wang et al., 2021), ANLI (Nie et al., 2020), and TextFooler (Jin et al., 2020). However, these approaches fundamentally differ from our proposed mathematical-level perturbations in purpose, methodology, and scope. Specifically, prior studies primarily focus on natural language understanding tasks, such as sentiment analysis and textual entailment, aiming to generate adversarial perturbations that cause misclassification without altering the ground truth. In contrast, our proposed perturbation method operates at a mathematical level in logical reasoning tasks and modifies *not only the problem,* *but also the ground-truth answer*. These mathematical perturbations ensure that the perturbed puzzle has a *distinctly different solution* compared to the original puzzle, while remaining superficially similar and maintaining a comparable difficulty level. This is guaranteed by the Perturber, Reasoner, and Solver components in our data generation framework. This approach provides a direct evaluation of the models' understanding of the underlying mathematical principles. By addressing logical reasoning robustness through mathematical-level perturbations, our work contributes a novel perspective distinct from prior studies.

B Details on K&K Benchmark

B.1 The Abstract Representation

We use a simple internal representation using basic Python primitives (integer, string and tuple) to encode each K&K puzzle. This allows easy inter-operation with the json format to simplify saving and loading. Specifically, for a N-people puzzle, each person is represented by the integer $0, \dots, N-1$. Each person's statement is represented by a tuple `(type, arguments, ...)`, where `type` indicate the statement type listed below:

- **Leaf Statements**: It can be either `('lying', i)` or `('telling-truth', i)`, where `i` is an integer and this statement assert the `i`th person is lying or truthful.
- **Composite Statements**: It can take one or more statements as arguments, and has the following types:
  - Negation `('not', statement)`
  - Conjunction `('and', statement1, statement2, ...)`
  - Disjunction `('or', statement1, statement2, ...)`
  - Implication `('->', statement1, statement2)`
  - Equivalence `('<=>', statement1, statement2)`

B.2 The Abstract Puzzle Module: Generator

The Generator samples a problem based on a random seed and a difficulty level specification (N, W, D), where N indicates the number of people, W indicates the max width of each statement, D indicates the max depth of each person's statement. To instantiate the problem, we initialize a random number generator, and sample a statement for each person sequentially. We sample each statement type uniformly at random. For composite statement with variable number of sub-statements, we also randomize the number according to the max width W. We restrict the sampling to only leaf statements if the max depth is exhausted. We avoid (skip and resample) some invalid (e.g., asserting self is lying) or uninteresting cases (e.g., a `and` statement with identical sub-statements).

The following is an example K&K puzzle with 5 people in the abstract representation. We will use this example to illustrate various component in the rest of the section.

Example puzzle of 5 people in the abstract representation

```
(('and', ('lying', 3), ('telling-truth', 4)),
 ('<=>', ('lying', 3), ('telling-truth', 4)),
 ('telling-truth', 4),
 ('telling-truth', 0),
 ('<=>', ('telling-truth', 2), ('lying', 2)))
```

B.3 The Abstract Puzzle Module: Solver and Reasoner

Each K&K problem can be transformed and solved as a Boolean satisfiability problem. Specifically, consider a puzzle involving N people, a possible solution assign a Boolean value to N variables $B_1, B_2, \dots, B_N$, where the truth value of B_i indicates whether the ith person is telling the truth. By definition, the ith person is telling the truth if and only if their statement S_i is true. Therefore, a valid solution to a K&K puzzle is a Boolean assignment for $B_1, B_2, \dots, B_N$ such that the following formula evaluates to true.

$$(B_1 \Leftrightarrow S_1) \wedge (B_2 \Leftrightarrow S_2) \wedge \cdots \wedge (B_N \Leftrightarrow S_N). \tag{2}$$

We implement our Solver and Reasoner based on this reduction. We take two different approaches here, because we want to find *all* possible solutions in the Solver, and we want to generate *intuitive* intermediate steps for the Reasoner.

Specifically, we are primarily interested in evaluating K&K puzzles with a unique valid solution. Therefore, we design our Solver to use a simple brute-force search that enumerates all possible Boolean assignments for N people and count the number of assignments that evaluate Eq. (2) to true. In our dataset construction, we only include puzzles whose solution count is exactly one.

In the Reasoner, we are interested in procedurally generating intermediate reasoning steps that lead to the final solution. We note that when explaining the reasoning steps for K&K puzzles, human or off-the-shelf LLMs rarely use the brute-force assignment search approach adopted in our Solver. Instead, they tend to examine the statement from each person sequentially, construct a *partial* assignment for the people examined so far, and backtrack when a contradiction is found. We design our Reasoner following the same procedure.

Specifically, we maintain a queue of people to be examined next, and a partial assignment of knight / knave for people that have been examined so far. In each step, we examine the next person from the queue by adding to the partial assignment the assumed knight / knave role for this person. Given the newly proposed assignment, we go through the known statements and check if there is a contradiction. (A) If a contradiction is found, we record the statement of contradiction as the explanation, and start backtracking. Backtracking will put people back into the to-be-examined queue until we reach a person who has an alternative unexamined role assignment. If no such person is found during backtracking, this means there is no valid solution for this problem. (B) If a contradiction is not found, we can proceed to examine the next person in the queue. Here we also implement a mechanism to reorder the queue so that it may match the human behavior better. For example, if the current person's statement is "If Noah is a knight, then Lily is a knave." then we would bring Noah and Lily to the front of the to-be-examined queue, provided that they are in the queue (i.e., have not been previously examined).

The reasoning steps are generated and stored using a similar format as the abstract representation of the puzzle as described in § B.1. The following snippet shows an example of the generated reasoning steps for the example puzzle shown above:

Example of generated reasoning steps in the abstract representation

```
[('proposal', {'assignment': True, 'outcome': 'ok', 'person': 0}),
 ('proposal', {'assignment': True, 'conflict_statement': (0, True), 'outcome': 'conflict', 'person': 3}),
 ('proposal', {'assignment': False, 'conflict_statement': (3, False), 'outcome': 'conflict', 'person': 3}),
 ('reconsider', {'exhausted': [3], 'person': 0}),
 ('proposal', {'assignment': False, 'outcome': 'ok', 'person': 0}),
 ('proposal', {'assignment': True, 'conflict_statement': (3, True), 'outcome': 'conflict', 'person': 3}),
 ('proposal', {'assignment': False, 'outcome': 'ok', 'person': 3}),
 ('proposal', {'assignment': True, 'conflict_statement': (0, False), 'outcome': 'conflict', 'person': 4}),
 ('proposal', {'assignment': False, 'outcome': 'ok', 'person': 4}),
 ('proposal', {'assignment': True, 'conflict_statement': (2, True), 'outcome': 'conflict', 'person': 2}),
 ('proposal', {'assignment': False, 'outcome': 'ok', 'person': 2}),
 ('proposal', {'assignment': True, 'conflict_statement': (1, True), 'outcome': 'conflict', 'person': 1}),
 ('proposal', {'assignment': False, 'outcome': 'ok', 'person': 1}),
 ('success', {'assignments': (False, False, False, False, False)})]
```

B.4 The Abstract Puzzle Module: Perturber

To support memorization measurement, the K&K Puzzle Perturber generate an perturbed version of a given puzzle. We design the perturbation with the following considerations:

- The perturbation should be "local", in the sense that the perturbed problem should be similar to the original problem when measured in some superficial ways, such as edit distance.
- The perturbation should keep the nature of the problem, i.e., the perturbed problem should be solvable using the same underlying principle, and the difficulty level should be roughly maintained.
- The perturbed puzzle should have a unique solution, which should be different from the solution of the original puzzle.

With those consideration, we support two different perturbations:

- Statement perturbation: randomly choose a person and sample a new statement tree for that person.

- Leaf perturbation: randomly choose a person, and from that person's existing statement tree, randomly choose one leaf node and change it.

B.5 The Natural Language Module: NL-Generator

The NL-Generator generate a K&K puzzle in natural language by converting a given abstract-form puzzle into the familiar text form described in natural language. For example, the puzzle presented (in the abstract representation) above can be materialized as follows:

Example puzzle converted to natural language representation

A very special island is inhabited only by knights and knaves. Knights always tell the truth, and knaves always lie. You meet 5 inhabitants: David, Zoey, Alexander, Aurora, and Isabella. In David's words: "Aurora is a knave and Isabella is a knight". Zoey expressed that Aurora is a knave if and only if Isabella is a knight. Alexander said, "Isabella is a knight." Aurora commented, "David is a knight". According to Isabella, "Alexander is a knight if and only if Alexander is a knave". So who is a knight and who is a knave?

Specifically, given a puzzle of N people in the abstract representation, our natural language representation generator first sample N human names, and then format each people's claim by plugging in the corresponding name mapping. We use some heuristics to make the conversion of the tree-structured logic statements to natural language sounds natural. Our current implementation randomly sample from 18 templates of making a statement claim and from the following common names — but this can be easily extended to include more.

```
COMMON_NAMES = ['Emma', 'Liam', 'Olivia', 'Noah', 'Ava', 'Ethan', 'Sophia',
                'Mason', 'Isabella', 'William', 'Mia', 'James', 'Charlotte',
                'Benjamin', 'Amelia', 'Lucas', 'Harper', 'Henry', 'Evelyn',
                'Alexander', 'Abigail', 'Michael', 'Emily', 'Daniel', 'Elizabeth',
                'Jacob', 'Sofia', 'Logan', 'Avery', 'Jackson', 'Ella', 'Sebastian',
                'Scarlett', 'Jack', 'Grace', 'Aiden', 'Chloe', 'Owen', 'Victoria',
                'Samuel', 'Riley', 'Matthew', 'Aria', 'Joseph', 'Lily', 'Luke',
                'Aurora', 'David', 'Zoey', 'Oliver', 'Penelope']
```

B.6 The Natural Language Module: NL-Reasoner

The NL-Reasoner generates detailed reasoning steps in natural language by converting the output from the abstract Reasoner to natural language descriptions using a similar approach as the NL-Generator. The following show the generated reasoning steps in natural language for the puzzle shown above:

Reasoning steps generated by the Reasoner

Let's think step by step, by considering whether each person is lying and if that leads to contradiction.

1. Assume David is a knight. No contradiction is found in their claim that Aurora is a knave and Isabella is a knight.
2. Aurora cannot be a knight, because this would contradict the claim of David that Aurora is a knave and Isabella is a knight.
3. Aurora cannot be a knave, because this would contradict the false claim of their own that David is a knight.
4. We have exhausted all possibilities for Aurora, so let us go back and reconsider David.
5. Assume David is a knave. No contradiction is found in their false claim that Aurora is a knave and Isabella is a knight.
6. Aurora cannot be a knight, because this would contradict the claim of their own that David is a knight.
7. Assume Aurora is a knave. No contradiction is found in their false claim that David is a knight.
8. Isabella cannot be a knight, because this would contradict the false claim of David that Aurora is a knave and Isabella is a knight.
9. Assume Isabella is a knave. No contradiction is found in their false claim that Alexander is a knight if and only if Alexander is a knave.
10. Alexander cannot be a knight, because this would contradict the claim of their own that Isabella is a knight.
11. Assume Alexander is a knave. No contradiction is found in their false claim that Isabella is a knight.
12. Zoey cannot be a knight, because this would contradict the claim of their own that Aurora is a knave if and only if Isabella is a knight.
13. Assume Zoey is a knave. No contradiction is found in their false claim that Aurora is a knave if and only if Isabella is a knight.

This leads to a feasible solution.

B.7 The Natural Language Module: NL-Perturber

The NL-Perturber generates perturbed puzzles at the language level. Note unlike in the perturbations generated by the abstract Perturber, NL-Perturber keep the underlying abstract puzzle intact and only modify the materialization in natural language. Therefore, the solution to the perturbed puzzle is identical to the solution to the original puzzle. Specifically, the NL-Perturber supports the following perturbations:

With those consideration in mind, we provide two family of perturbations:

- Uncommon name: replace the names of the characters with randomly sampled names from the set of uncommon names.
- Random role: change the role name from knight/knave to other pairs of role names. To avoid introducing bias, we sample from pairs of good/bad role names, including "*saint/sinner, hero/villain, angel/devil, altruist/egoist, sage/fool, pioneer/laggard*".
- Reorder statement: shuffle the order of presenting each person's statement.
- Flip role: change the role from knight/knave to knave/knight, i.e., knave will be telling the truth while knight will be lying.

The uncommon names are sampled from the following list:

```
UNCOMMON_NAMES = [
  'Zephyr', 'Elowen', 'Caspian', 'Isolde', 'Osiris', 'Vesper', 'Thaddeus', 'Ondine',
  'Lysander', 'Xanthe', 'Oberon', 'Calliope', 'Leander', 'Eulalia', 'Florian', 'Forsythe',
  'Nephele', 'Peregrine', 'Ianthe', 'Lazarus', 'Elodie', 'Cillian', 'Ottoline', 'Evander',
  'Saffron', 'Caius', 'Zora', 'Cyprian', 'Amaryllis', 'Theron', 'Perdita', 'Ignatius',
  'Zephyrine', 'Balthazar', 'Melisande', 'Zinnia', 'Sylvester', 'Cosima', 'Leocadio',
  'Percival', 'Oceane', 'Evanthe', 'Zenobia', 'Eurydice', 'Quillan', 'Aeronwen',
  'Thorsten', 'Xiomara', 'Zephyrus', 'Ysolde'
]
```

Note the flip role perturbation is somewhat adversarial as it goes against the common intuition that good role tends to tell the truth while bad role tends to lie. We indeed observe that the models would make a lot of mistakes under this perturbation, despite that the perturbed problem is perfect valid and unambiguous. However, the study of how model's bias impact its reasoning capability is not the main focus of this paper. So we keep this perturbation as reference but primarily focus on "benign" perturbations.

B.8 Dataset Generation

K&K dataset During our data construction, we use the maximum width $W = 2$ and depth $D = 2$, and the number of persons in the puzzle $N = 2, 3, 4, 5, 6, 7, 8$.

We present the length distributions of K&K training dataset in Fig. 17. The length distributions of the test dataset are similar to those of the training dataset.

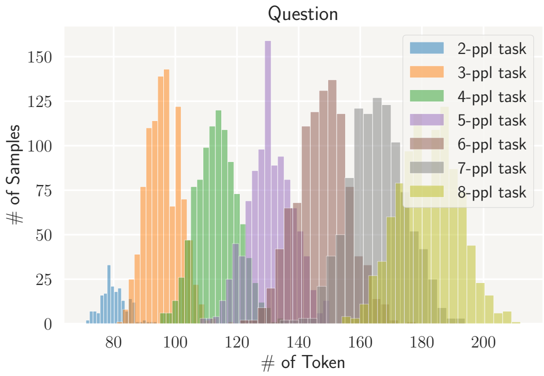


Figure 12: K&K questions

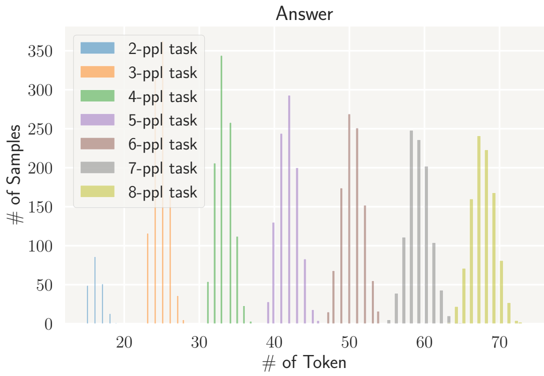


Figure 13: K&K answers

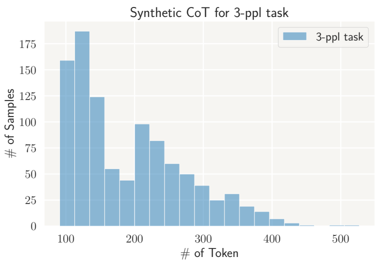


Figure 14: 3-people K&K synthetic CoTs

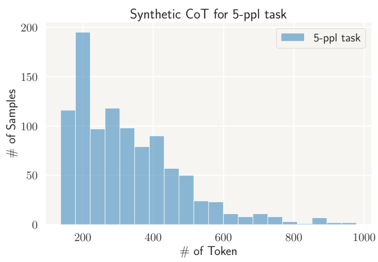


Figure 15: 5-people K&K synthetic CoTs

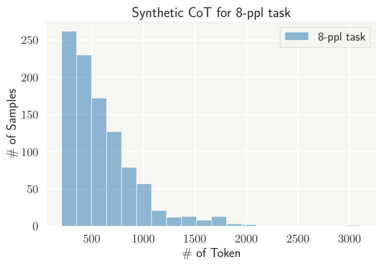


Figure 16: 8-people K&K synthetic CoTs

Figure 17: Length distributions of K&K training data.

Local perturbation Tab. 1 presents the example knight (truth-teller) and knave (liar) scenario involving two people: Liam and Aria, with corresponding logical statements, and converted English statements, questions, and answers. It also shows three versions of the problems: an original example, a leaf-perturbed version, and a statement-perturbed version. Specifically, (1) leaf perturbation changes a "leaf" in the logical tree - a single truth value. In this case, it flipped Jacob's status in Oliver's statement from knave (liar) to knight (truth-teller) (2) Statement perturbation changes the entire structure of a statement. Here, it changed Oliver's statement entirely. Both perturbations result in changing the answer. The leaf perturbation creates a subtle change in one statement that flips the logical outcome, while the statement perturbation changes the entire one statement.

Moreover, we compare the math-level perturbation with language-level perturbation in Fig. 18.

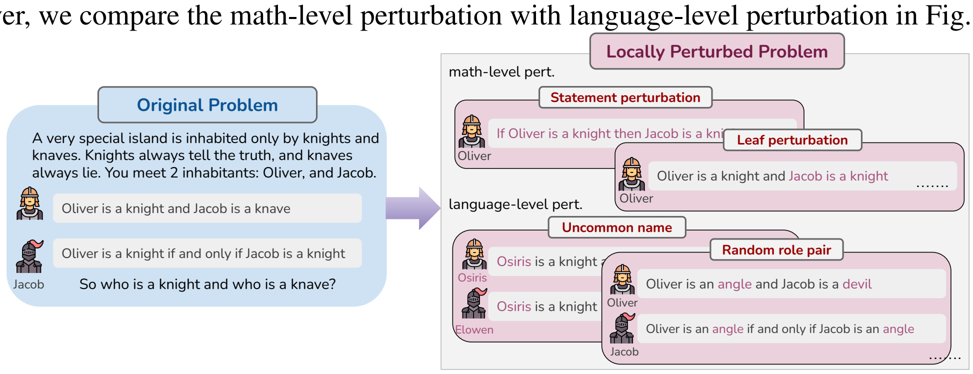


Figure 18: Comparison between different locally perturbed problems.

Table 1: 2-person puzzle generation with the knight (telling-truth) and knave (lying) and comparison between original sample, leaf-perturbed sample, and statement-perturbed sample.

Type	Example	Leaf Perturbed Example	Statement Perturbed Example
person	Oliver (person index 0), Jacob (person index 1)		
logical statement	Oliver: ('and', ('telling-truth', 0), ('lying', 1)) Jacob: ('⇔', ('telling-truth', 0), ('telling-truth', 1))	Oliver: ('and', ('telling-truth', 0), **('telling-truth', 1)**) Jacob: ('⇔', ('telling-truth', 0), ('telling-truth', 1))	Oliver: **('→', ('telling-truth', 0), ('telling-truth', 1))** Jacob: ('⇔', ('telling-truth', 0), ('telling-truth', 1))
English statement	Oliver: Oliver is a knight and Jacob is a knave Jacob: Oliver is a knight if and only if Jacob is a knight	Oliver: Oliver is a knight and **Jacob is a knight** Jacob: Oliver is a knight if and only if Jacob is a knight	Oliver: **If Oliver is a knight then Jacob is a knight** Jacob: Oliver is a knight if and only if Jacob is a knight
question	A very special island is inhabited only by knights and knaves. Knights always tell the truth, and knaves always lie. You meet 2 inhabitants: Oliver, and Jacob. Oliver commented, "Oliver is a knight and Jacob is a knave". Jacob remarked, "Oliver is a knight if and only if Jacob is a knight". So who is a knight and who is a knave?	A very special island is inhabited only by knights and knaves. Knights always tell the truth, and knaves always lie. You meet 2 inhabitants: Oliver, and Jacob. Oliver commented, "Oliver is a knight and Jacob is a knight". Jacob remarked, "Oliver is a knight if and only if Jacob is a knight". So who is a knight and who is a knave?	A very special island is inhabited only by knights and knaves. Knights always tell the truth, and knaves always lie. You meet 2 inhabitants: Oliver, and Jacob. Oliver commented, "If Oliver is a knight then Jacob is a knight". Jacob remarked, "Oliver is a knight if and only if Jacob is a knight". So who is a knight and who is a knave?
answer	(1) Oliver is a knight (2) Jacob is a knave	(1) Oliver is a knight (2) **Jacob is a knight**	(1) Oliver is a knight (2) **Jacob is a knight**

As mentioned in § 2, the Perturber of the abstract puzzle module generates a perturbed puzzle with a *unique* solution that is *different* from the original puzzle, or until the maximum number of attempts is reached. We set this limit to 2000 attempts.

- For statement perturbation, the Perturber can always return a valid perturbed puzzle due to the large perturbation space.
- For leaf perturbation, since the process is restricted to a single leaf node, it may not always find a valid perturbed puzzle within the constraints of unique and different solution. Below are the detailed proportions of valid leaf perturbations on training samples (under 2000 max attempts for each sample): 76% valid for 2-person task; 93.4% valid for 3-person task; 95.4% valid for 4-person task; 98.8% valid for 5-person task; 99.5% valid for 6-person task; 100% valid for 7/8-person tasks.

C Details on DynamicZebra

To extend our approach to more tasks and validate our findings, we designed a new dynamic data generation pipeline for a class of structured logic puzzles, Zebra puzzles (also known as Einstein's puzzles (Zebra Puzzle)). These puzzles are representative of constraint satisfaction problems and are commonly used to evaluate logical reasoning. While Lin et al. (2024) proposed ZebraLogic containing 1000 such puzzles, it does not include the generation algorithm, so it is not suitable for controlled memorization experiments. Our contribution lies in building a new dataset, which we name DynamicZebra, with fully controlled generation and perturbation mechanisms.

We provide details of the dataset:

- Training set: We generate a total of 5000 Zebra puzzles with 5 difficulty levels: 4×4, 4×5, 4×6, 5×4, 5×5, where the numbers represent *n_attributes* × *n_houses*. These samples are used for mixed-difficulty training.
- Perturbed training set: For each training sample, we construct 2 perturbed variations. A perturbed version maintains the same solution grid except that we select a random row (attribute) and swap the values of two randomly selected houses. The puzzle description remains nearly the same, except that

clues involving the swapped values are updated accordingly. This allows us to isolate whether the model memorizes specific value positions or can reason about the solution structure.

- Test set: We generate 100 samples for each of 12 test difficulties, ranging from 3×3 to 6×6. These are used to evaluate generalization to unseen problem variants.

D Experimental Setups

D.1 Models

Tab. 2 provides the details of the models evaluated in our study.

Table 2: HuggingFace links or endpoint specifications for evaluated models.

Model	Link
Llama3-8B	https://huggingface.co/meta-llama/Meta-Llama-3-8B
Phi-3-mini	https://huggingface.co/microsoft/Phi-3-mini-4k-instruct
Phi-3-medium	https://huggingface.co/microsoft/Phi-3-medium-4k-instruct
NuminaMath-7B-CoT	https://huggingface.co/AI-MO/NuminaMath-7B-CoT
Deepseek-Math-7B	deepseek-ai/deepseek-math-7b-instruct
Claude-3.5-Sonnet	https://www.anthropic.com/news/claude-3-5-sonnet, claude-3-5-sonnet-20240620 endpoint
GPT4o-mini	https://platform.openai.com/docs/models/, gpt-4o-mini-2024-07-18 endpoint
GPT4o	https://platform.openai.com/docs/models/, gpt-4o-2024-05-13 endpoint
Gemini-1.5-Flash-002	https://console.cloud.google.com/vertex-ai/model-garden, gemini-1.5-flash-002 endpoint
Gemini-1.5-Prof-002	https://console.cloud.google.com/vertex-ai/model-garden, gemini-1.5-pro-002 endpoint

D.2 Experimental Details

D.2.1 Evaluation

By default, we utilize zero-shot direct prompting with task-specific instructions for open-ended question-answering. We employ the following prompt:

0-shot Direct Prompting

Your task is to solve a logical reasoning problem. You are given set of statements from which you must logically deduce the identity of a set of characters.

You must infer the identity of each character. At the end of your answer, you must clearly state the identity of each character by following the format:

CONCLUSION:
(1) ...
(2) ...
(3) ...

Question: {question}
Answer:

In addition to the 0-shot direct prompting used in the main paper, we explore 0-shot Chain of Thought (CoT) prompting and 1-shot direct/CoT prompting and report the results in Appendix § E.

0-shot CoT Prompting

Your task is to solve a logical reasoning problem. You are given set of statements from which you must logically deduce the identity of a set of characters.

You must infer the identity of each character. First, explain your reasoning. At the end of your answer, you must clearly state the identity of each character by following the format:

CONCLUSION:
(1) ...
(2) ...
(3) ...

Question: {question}
Answer: Let's think step by step

In addition, we utilize a specific CoT prompting format for instruction-tuned models: DeepSeek-Math-7B and NuminaMath-7B-CoT, as recommended by their developers:

Please reason step by step, and put your final answer within \boxed{}.

This replaces the previous prompt, "Let's think step by step."

1-shot Direct Prompting

Your task is to solve a logical reasoning problem. You are given set of statements from which you must logically deduce the identity of a set of characters.

You must infer the identity of each character. At the end of your answer, you must clearly state the identity of each character by following the format:

CONCLUSION:
(1) ...
(2) ...
(3) ...

Question: A very special island is inhabited only by knights and knaves. Knights always tell the truth, and knaves always lie. You meet 2 inhabitants: Jack, and Sophia. Jack tells you that Sophia is not a knave. Sophia says that If Jack is a knight then Sophia is a knight. So who is a knight and who is a knave?
Answer:
CONCLUSION:
(1) Jack is a knight
(2) Sophia is a knight

Question: {question}
Answer:

1-shot CoT Prompting

Your task is to solve a logical reasoning problem. You are given set of statements from which you must logically deduce the identity of a set of characters.

You must infer the identity of each character. First, explain your reasoning. At the end of your answer, you must clearly state the identity of each character by following the format:

CONCLUSION:
(1) ...
(2) ...
(3) ...

Question: A very special island is inhabited only by knights and knaves. Knights always tell the truth, and knaves always lie. You meet 2 inhabitants: Ella, and Penelope. In a statement by Ella: "Ella is a knight or Penelope is a knight". According to Penelope, "Ella is a knave if and only if Penelope is a knight". So who is a knight and who is a knave?
Answer: Let's think step by step, by considering whether each person is lying and if that leads to contradiction. Assume Ella is a knight. Penelope cannot be a knight, because this would contradict the claim of their own. Penelope cannot be a knave, because this would contradict the false claim of their own. We have exhausted all possibilities for Penelope, so let us go back and reconsider Ella. Assume Ella is a knave. Penelope cannot be a knight, because this would contradict the false claim of Ella. Assume Penelope is a knave. This leads to a feasible solution.
CONCLUSION:
(1) Ella is a knave
(2) Penelope is a knave

Question: {question}
Answer: Let's think step by step

In our evaluation process, we use greedy decoding with temperature $t = 0$ for all models and a maximum token length of 2048.

To assess the correctness, we implement keyword matching: a response is considered correct if each person's ground truth identity appears in the conclusion part of the model's output.

D.2.2 Fine-tuning

Training instance Each training instance in Direct FT includes the task instruction, question, and the correct conclusion. In CoT FT, each training instance includes the task instruction, question, synthetic reasoning steps, and the correct conclusion. Both formats are structured similarly to task instructions followed by a single demonstration used in 1-shot Direct Prompting or 1-shot CoT Prompting.

Training loss In Direct FT, the loss for each training instance is computed on the tokens that appear directly after "### Answer:\n". In CoT FT, the loss is calculated on the tokens that appear directly after "### Answer: Let's think step by step".

Training hyperparameters For Llama3-8B fine-tuning, we used LoRA fine-tuning with the following standard hyperparameters: a batch size of 4, gradient accumulation steps of 8, and 5e-5 learning rate. The LoRA configuration was set as follows: rank $r = 32$, scaling factor $\alpha = 32$, and dropout rate 0.05. No quantization techniques were used. We fine-tune for a maximum of 100 epochs. We primarily reported results before 50 epochs, as we found the model typically converged by then.

For GPT4o-mini fine-tuning, we utilized the default hyperparameters provided by the OpenAI fine-tuning API. The model was fine-tuned for 5 epochs to achieve high accuracy within reasonable budget.

Reported Training accuracy For GPT4o-mini, the training accuracy for each N-people K&K task is calculated using 100 training samples due to budget constraints on API usage. For open-source Llama3-8B, the training accuracy is based on the full set of training samples.

D.2.3 Probing

As described in § 4.2, in the probing experiments, we train logistic regression models on the model's intermediate outputs from different transformer blocks, to distinguish between correct and incorrect statements. For each transformer block, we extract the MLP layer's output.

The correct/incorrect statements consist of a K&K puzzle and a conclusion about a character's role in the puzzle. For example, considering the following 2-people K&K puzzle:

> A very special island is inhabited only by knights and knaves. Knights always tell the truth, and knaves always lie. You meet 2 inhabitants: Oliver, and Ethan. Oliver told you that Oliver is a knight or Ethan is a knave. In a statement by Ethan: "Oliver is a knight". So who is a knight and who is a knave?

with the correct answer being

> Oliver is a knight, and Ethan is a knight.

We can generate two correct statements:

- A very special island is inhabited only by knights and knaves. Knights always tell the truth, and knaves always lie. You meet 2 inhabitants: Oliver, and Ethan. Oliver told you that Oliver is a knight or Ethan is a knave. In a statement by Ethan: "Oliver is a knight". So who is a knight and who is a knave? **Oliver is a knight.**
- A very special island is inhabited only by knights and knaves. Knights always tell the truth, and knaves always lie. You meet 2 inhabitants: Oliver, and Ethan. Oliver told you that Oliver is a knight or Ethan is a knave. In a statement by Ethan: "Oliver is a knight". So who is a knight and who is a knave? **Ethan is a knight.**

And two incorrect statements:

- A very special island is inhabited only by knights and knaves. Knights always tell the truth, and knaves always lie. You meet 2 inhabitants: Oliver, and Ethan. Oliver told you that Oliver is a knight or Ethan is a knave. In a statement by Ethan: "Oliver is a knight". So who is a knight and who is a knave? **Oliver is a knave.**
- A very special island is inhabited only by knights and knaves. Knights always tell the truth, and knaves always lie. You meet 2 inhabitants: Oliver, and Ethan. Oliver told you that Oliver is a knight or Ethan is a knave. In a statement by Ethan: "Oliver is a knight". So who is a knight and who is a knave? **Ethan is a knave.**

D.2.4 Distinguishing Memorization from Reasoning

For GPT4o-mini and Llama3-8B, we calculate the memorization score for each training sample within each complete N-people K&K training dataset. As discussed in § 6, we omit samples where $\mathsf{Acc}(f;x) = 0$ and label the remaining samples based on whether they are consistently solved under perturbation. We then split the dataset into 80%/20% train/test sets and perform binary classification.

D.2.5 Computation Resources

The fine-tuning experiments are conducted on 2 NVIDIA A100 GPU cards, each with 80GB of memory. The LLM evaluation experiments can be conducted on one NVIDIA RTX A6000 GPU card with 48 GB of memory.

E Additional Experimental Results

E.1 Benchmark Performance of Off-the-shelf Models

Off-the-shelf models We evaluate Llama3-8B, Phi-3-mini, Phi-3-medium, NuminaMath-7B-CoT, and Deepseek-Math-7B using 0/1-shot Direct/CoT prompting in Fig. 19. The results indicate that these open-source models exhibit poor accuracy on K&K tasks, particularly as the number of people in the K&K puzzles increases. Different prompting methods do not significantly enhance performance.

Moreover, we evaluate GPT4o-mini under the self-consistency (Wang et al., 2023b) where we query each puzzle 40 times under temperature 0.7. Tab. 3 shows that self-consistency provides limited improvement on the 3-ppl task and fails to enhance performance on the more challenging 8-ppl task, suggesting that the model fundamentally struggles with solving such complex problems.

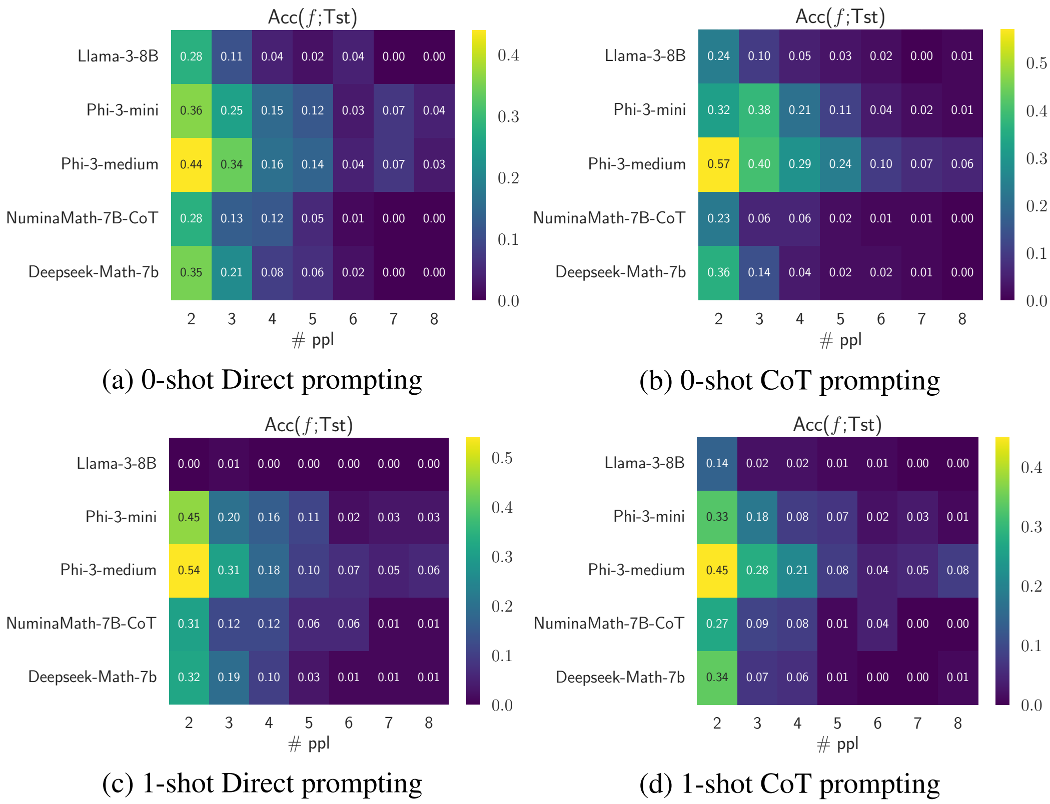


Figure 19: Acc(f; Tst) and LiMem(f; Tst) of off-the-shelf models under various prompt formats.

Table 3: Self-consistency (Wang et al., 2023b) can enhance the accuracy of GPT4o-mini on the easy 2-ppl K&K task, but has limited improvement on 3-ppl task and fails to help on the challenging 8-ppl task, which suggests that the model cannot fundamentally solve such complex problems.

Method	Test Accuracy		
	2-ppl	3-ppl	8-ppl
Direct Prompting	0.63	0.42	0.01
Direct Prompting + Self-consistency	0.74	0.43	0.02

E.2 Memorization Measurement

Fine-tuned models As shown in Fig. 20, the inconsistency ratio on correctly solved training puzzles (y-axis) of CoT-FTed or Direct-FTed GPT4o-mini tends ot decreases over the training epochs (x-axis), despite that the memorization score $\mathsf{LiMem}(f; \mathsf{Tr})$ on training samples also increases (i.e., a larger proportion of memorized samples in the training set). The memorization score $\mathsf{LiMem}(f; \mathsf{Tr})$ under role-flipping is significantly higher than other perturbation, possibly due to an internal bias that knights are truthful.

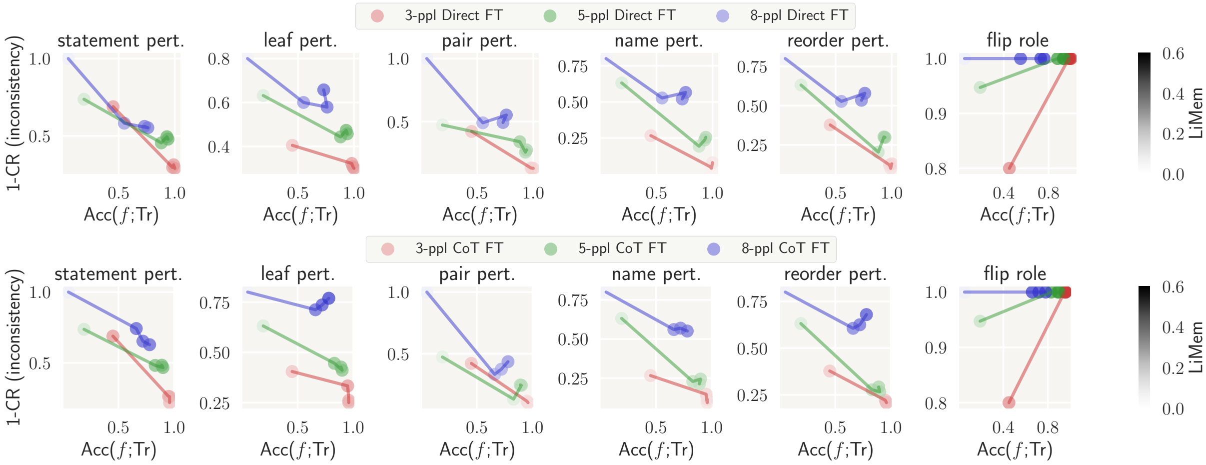


Figure 20: Inconsistency ratio (y-axis) of fine-tuned GPT4o-mini (first row: Direct FT; second row: CoT FT) decreases over training epochs (x-axis), despite that the memorization becomes stronger as reflected by larger $\mathsf{LiMem}(f; \mathsf{Tr})$ (deeper color).

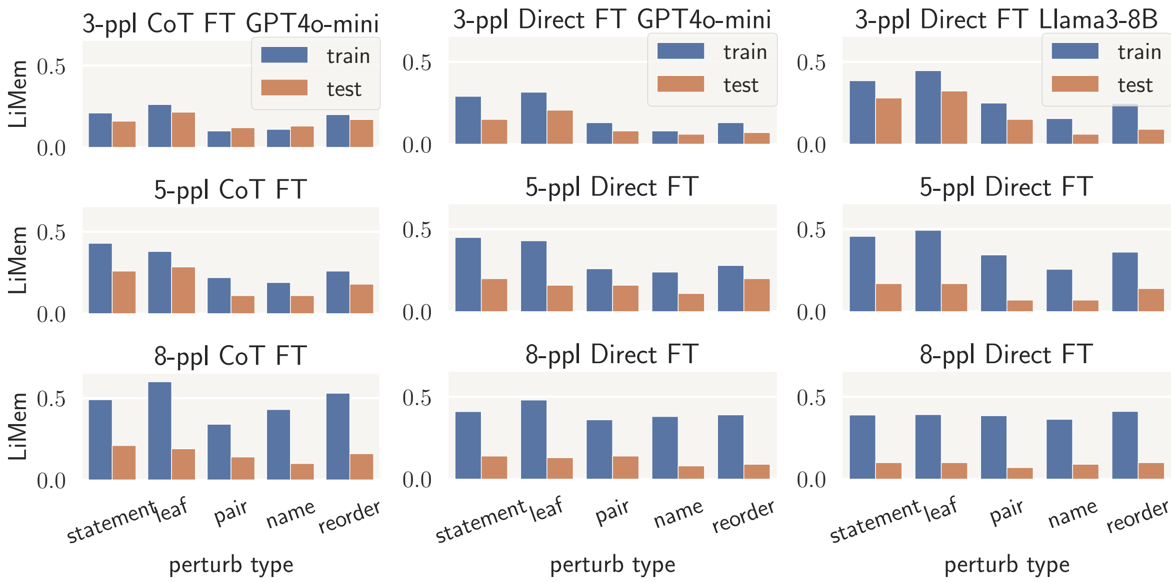


Figure 21: Fine-tuned LLMs exhibit high memorization score on the training set under different perturbations, especially for hard tasks. The score on the test set can be smaller than on the training set. Models show stronger memorization under math-level perturbations compared to language-level perturbations.

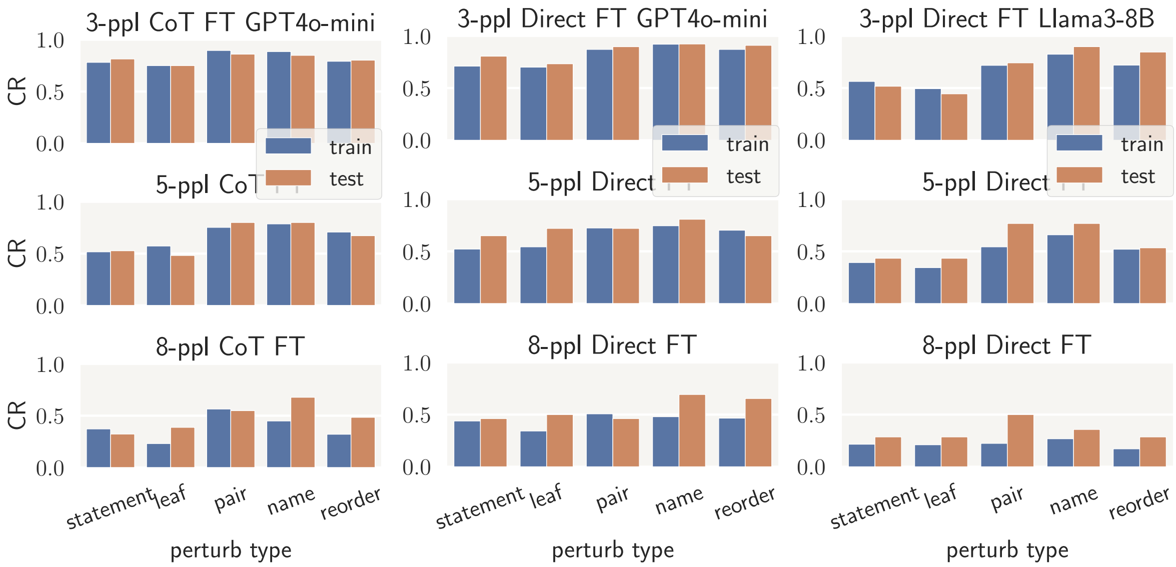


Figure 22: Consistency Ratio (CR ↑) under local perturbations. Fine-tuned LLMs generally demonstrate a higher consistency ratio on solved problems in the test set compared to the train set, particularly for challenging tasks such as 5/8-person puzzles. On the 3-person puzzle task, the consistency ratio between the train and test sets remains comparable. The consistency ratio generally is higher in easy tasks than in hard tasks.

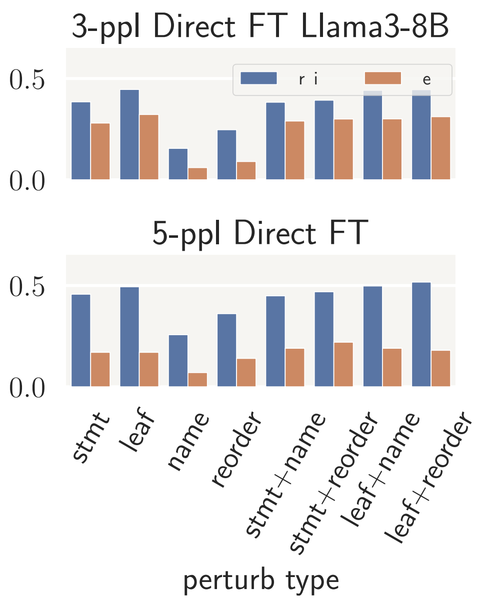


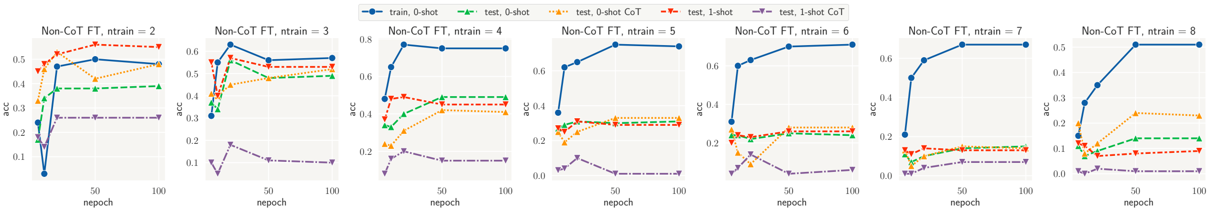
Figure 23: Memorization scores of Directly Fine-Tuned Llama3-8B under various math-level (statement, leaf) and language-level (name, reorder) perturbations. Combining math-level and language-level perturbations progressively can result in higher memorization scores (e.g., leaf + reorder), especially compared to applying language-level perturbations alone.

E.3 Evaluation on Reasoning Capability

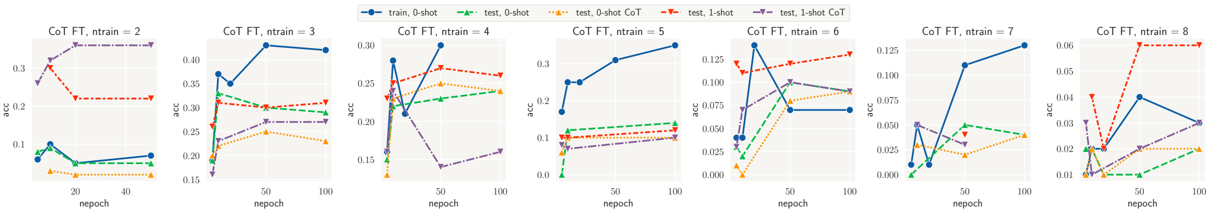
E.3.1 Llama3-8B

Accuracy over epochs Fig. 24 reports the train and test accuracy (under different evaluation configurations) for the Llama3-8B model fine-tuned on N-person tasks across multiple training epochs.

Transferability We present the transferability results for the K&K task across different problem sizes and training epochs in Fig. 25 and Fig. 26. Fig. 25 shows the accuracy improvements relative to the baseline with no fine-tuning, while Fig. 26 reports the absolute accuracy values.



(a) Direct FT



(b) CoT FT

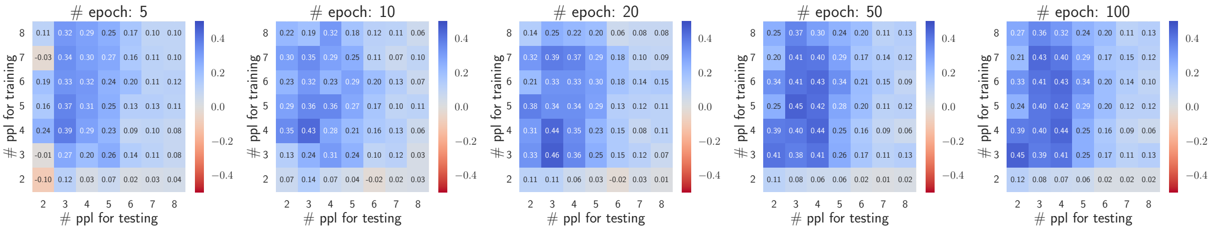
Figure 24: Train and test accuracy (under different evaluation configurations) for the Llama3-8B model fine-tuned on N-person tasks across multiple training epochs.

Fine-tuning on 10k 8-people K&K samples Fig. 9 shows that **(1)** $10k$-FT significantly outperforms $1k$-FT across all tasks, reaching $\sim 90\%$ test accuracy on moderately difficult 4/5-people puzzles. **(2)** CoT FT is generally more effective than Direct FT with $10k$ samples, likely due to the guidance provided by reasoning steps. **(3)** An exception is the 2-people task, where the training and testing distribution gap causes the CoT FTed model to occasionally get stuck in a loop of listing assumptions and contradictions, resulting in long, repetitive responses without reaching a conclusion[5]. **(4)** Direct FT with $10k$ puzzles achieves surprisingly high test accuracy on all tasks, e.g., 52% on 8-people tasks, where the un-FTed model scores near 0. Notably, the models do not see reasoning steps during training and rely solely on memorizing answers. We also observe high transferability for $10k$ Direct FTed Llama3-8B in Fig. 27, e.g., 87% test accuracy on 3-people puzzles.

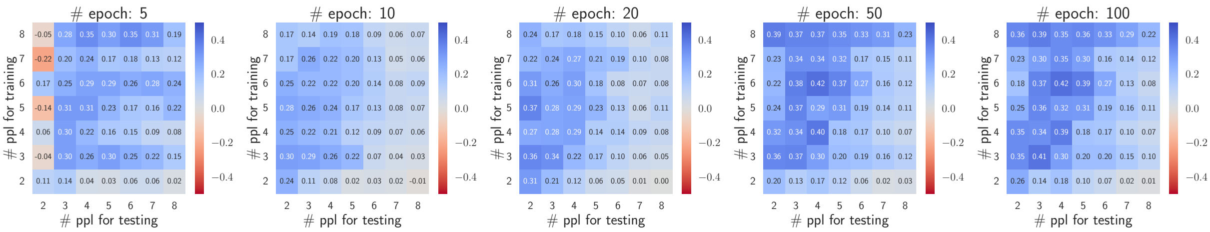
The results in Fig. 27 shows that $10k$ fine-tuning achieves significantly higher test accuracy than $1k$ fine-tuning on all tasks. Direct FT with $10k$ puzzles shows surprisingly high test accuracy, e.g., 87% accuracy on 3-person tasks, where the un-FTed model has nearly 0 accuracy as shown in Fig. 3. Notably, the models don't see reasoning steps during training and rely solely on memorizing answers. It also suggests that training on the hardest (8-person) tasks helps the model learn certain underlying rules that can be transferred to solve easier tasks.

However, the test accuracy drops for Llama3-8B when Direct FTing on $10k$ samples for overly long epochs, especially evaluated on 2-people K&K task, potentially due to overfitting to the more complicated 8-people training task.

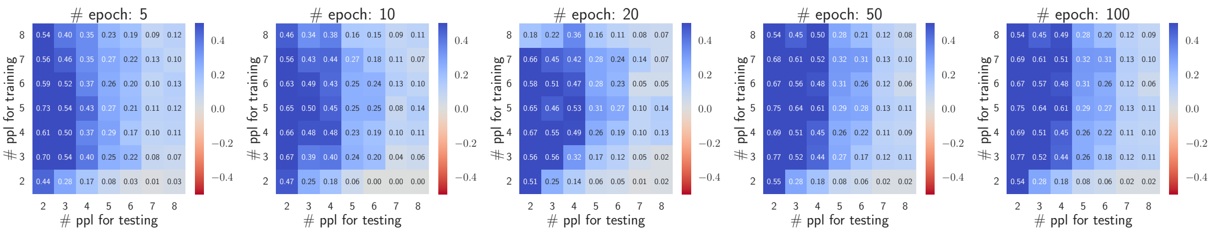
[5]We observe similar accuracy drop on 2-people task for Llama3-8B (see Fig. 27) when it is Direct FTed for overly long epochs. See more examples and discussions in § E.3.2.



(a) 0-shot Direct Prompting

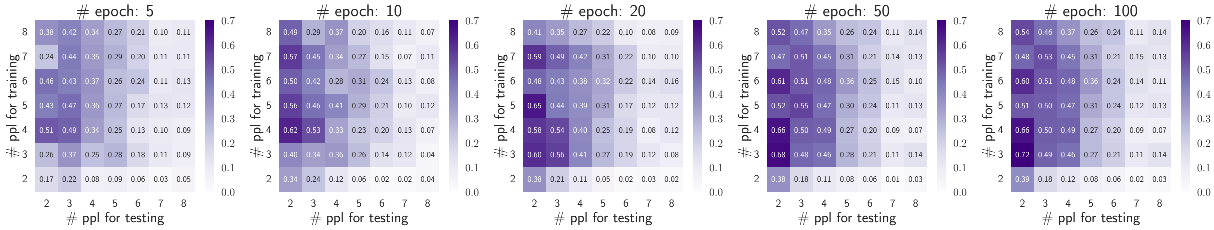


(b) 0-shot CoT Prompting

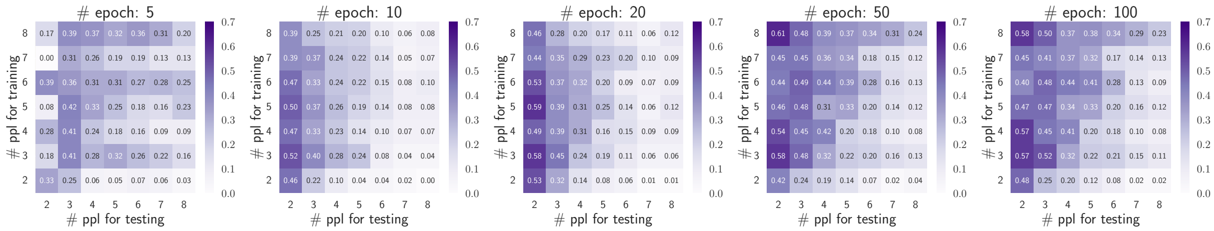


(c) 1-shot Direct Prompting

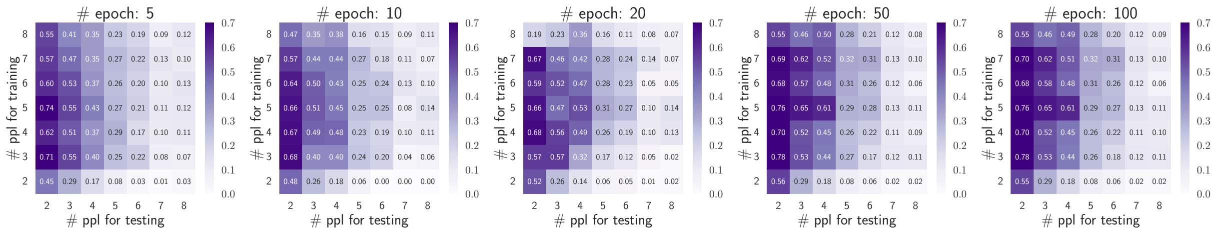
Figure 25: Improvement in test accuracy on N-person problems for Llama3-8B fine-tuned on M-person problems **with direct FT**, compared to the unfine-tuned model, under various evaluation configurations.



(a) 0-shot Direct Prompting



(b) 0-shot CoT Prompting



(c) 1-shot Direct Prompting

Figure 26: Test accuracy on N-person problems for Llama3-8B fine-tuned on M-person problems **with direct FT**, under various evaluation configurations.

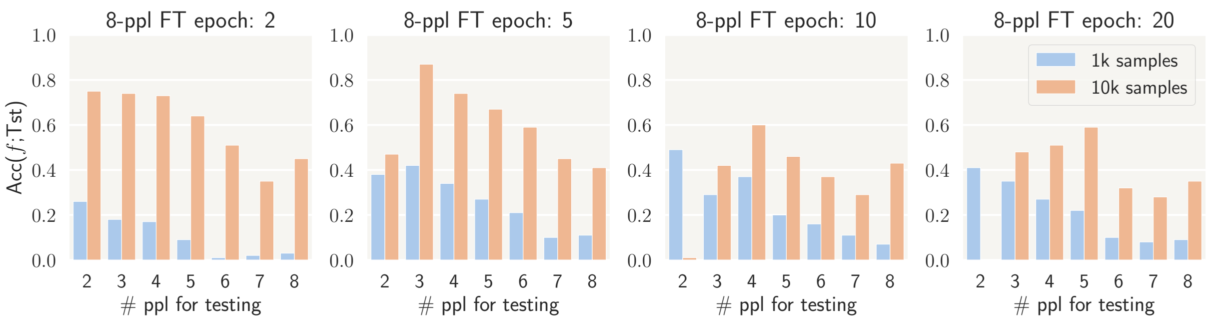
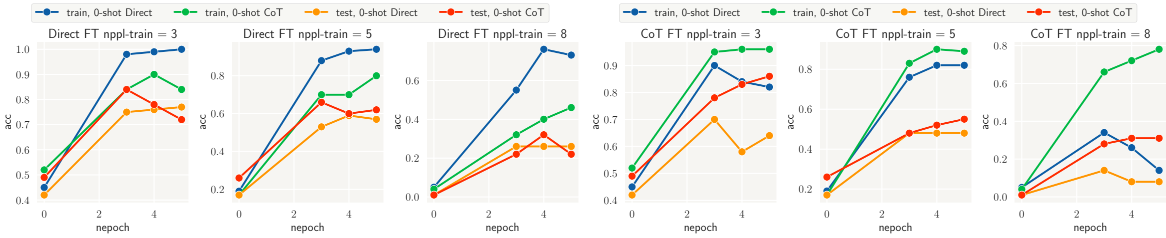


Figure 27: Transferability of Llama3-8B Direct-FTed on $1k$/$10k$ samples at different epochs.

E.3.2 GPT4o-mini

Accuracy over epochs Fig. 28 reports the train and test accuracy (under different evaluation configurations) for the GPT4o-mini model fine-tuned on N-person tasks across multiple training epochs.

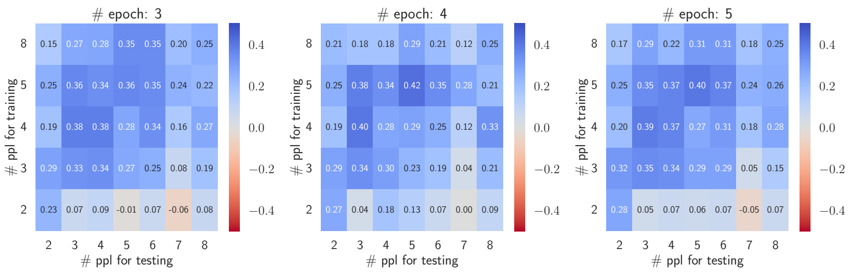
Using the same paradigm for training and evaluation (i.e., Direct FT & Direct Prompting, CoT FT & CoT Prompting) usually achieves the best accuracy for GPT4o-mini on training dataset and test dataset. We focus on 0-shot setting for GPT4o-mini evaluation given its stronger capacity and higher accuracy than Llama3-8B.



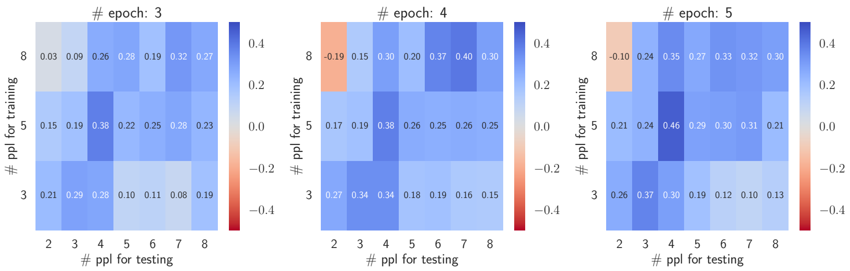
(a) Direct FT (b) CoT FT

Figure 28: Train and test accuracy (under different evaluation configurations) for the GPT4o-mini model fine-tuned on N-person tasks across multiple training epochs.

Transferability We present the transferability results for the K&K task across different problem sizes and training epochs in Fig. 29 and Fig. 30. Fig. 29 shows the accuracy improvements relative to the baseline with no fine-tuning, while Fig. 30 reports the absolute accuracy values.



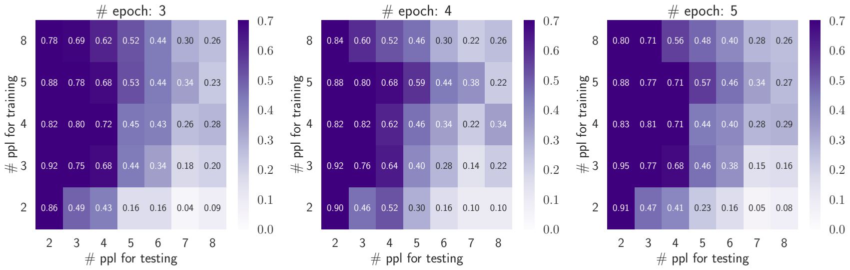
(a) Direct FT & 0-shot Direct Prompting



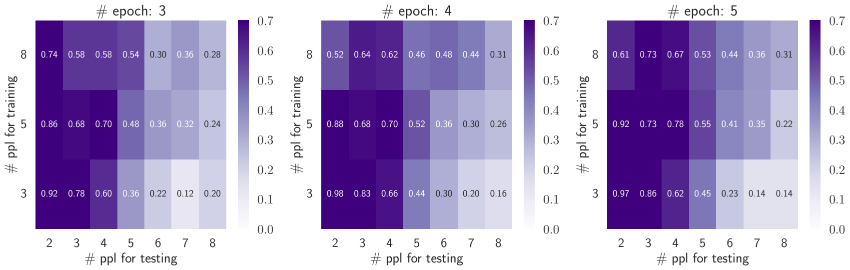
(b) CoT FT & 0-shot CoT Prompting

Figure 29: Improvement in test accuracy on N-person problems for GPT4o-mini fine-tuned on M-person problems, under two finetuning/evaluation configurations.

Fine-tuning on 10k 8-people K&K samples We present the transferability results with absolute test accuracy for the K&K task across different 8-people task training sizes and training epochs in Fig. 31. As shown, GPT4o-mini achieves high accuracy on all tasks at early epochs (e.g., 3 epochs). We also find that GPT4o-mini exhibits poor test accuracy on two-person testing puzzles when CoT-FTed on 10k 8-people puzzles, unlike the Direct FTed model that have stable performance across all task. In the failure case below, the CoT-FTed GPT4o-mini gets stuck in a loop of listing assumptions and contradictions, resulting in long, repetitive responses without reaching a conclusion.

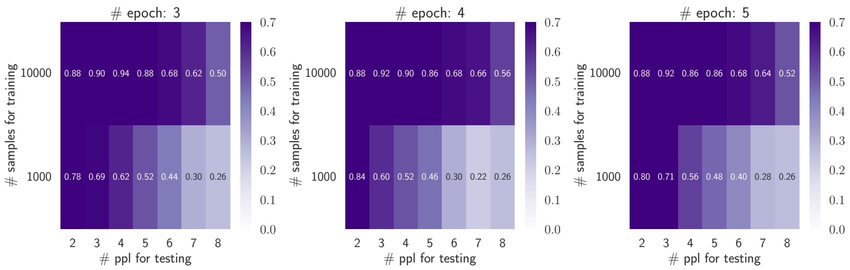


(a) Direct FT & 0-shot Direct Prompting

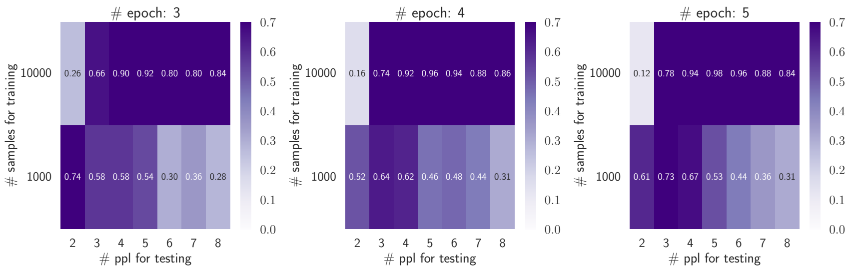


(b) CoT FT & 0-shot CoT Prompting

Figure 30: Test accuracy on N-person problems for GPT4o-mini fine-tuned on M-person problems, under two finetuning/evaluation configurations.



(a) Direct FT & 0-shot Direct Prompting



(b) CoT FT & 0-shot CoT Prompting

Figure 31: Test accuracy on N-person problems for GPT4o-mini fine-tuned on $1k$/$10k$ 8-people problems, under two finetuning/evaluation configurations.

Failure case of GPT4o-mini CoT FTed on 10k 8-people K&K task

{omitted task instruction}
Question: A very special island is inhabited only by knights and knaves. Knights always tell the truth, and knaves always lie. You meet 2 inhabitants: Logan, and Olivia. Logan was heard saying, "Logan is a knight". "If Olivia is a knight then Logan is a knight" - Olivia. So who is a knight and who is a knave?
Answer: Let's think step by step

// response from GPT4o-mini
by considering whether each person is lying and if that leads to contradiction. Assume Logan is a knight. No contradiction is found in their claim that Logan is a knight. Assume Olivia is a knight. No contradiction is found in their claim that If Olivia is a knight then Logan is a knight. Assume this leads to no contradiction. No contradiction is found in their claim that If Olivia is a knight then Logan is a knight. This leads to no contradiction in their claim that Logan is a knight. This leads to no contradiction in their claim that If Olivia is a knight then Logan is a knight. This leads to no contradiction in their claim that Logan is a knight. This leads to no contradiction in their claim that If Olivia is a knight then Logan is a knight. This leads to no contradiction in their claim that If Olivia is a knight then Logan is a knight. This leads to no contradiction in their claim that Logan is a knight. This leads to no contradiction in their claim that If Olivia is a knight then Logan is a knight. This leads to no contradiction in their claim that Logan is a knight.

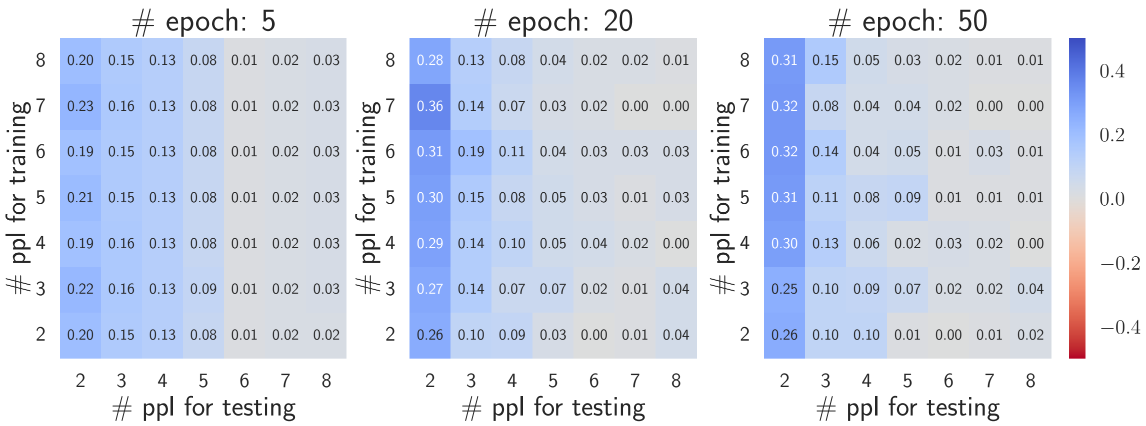


Figure 32: Test accuracy improves on N-people puzzles for Llama3-8B Direct FTed on M-people puzzles *with completely wrong answers*, compared to the unfine-tuned model. This evaluation uses 1-shot direct prompting (see Fig. 34 for results under different prompting setups).

E.4 Direct FT with Wrong Answers

To further explore what could the models learn from the question-answer pairs without detailed reasoning steps, we consider an extreme scenario of learning with incorrect answers: for each N-people training puzzle, we randomly select $\tilde{N}$ from $[1, N]$ and flip the knight/knave identities of $\tilde{N}$ randomly chosen individuals. Surprisingly, Fig. 32 shows that Direct FT with incorrect answers still leads to non-trivial improvements for Llama3-8B. These improvements occur gradually over more epochs, suggesting that the model progressively developed reasoning skills during fine-tuning. Fig. 34, Fig. 35 and Fig. 36 show the results of Direct FT with 100%, 75% and 50% incorrect answers for the Llama3-8B model across different prompting setups. Consistent with our earlier findings in § E.4, fine-tuning with incorrect answers still significantly improves K&K performance, especially with 0-shot CoT prompting or 1-shot direct prompting.

Note that in this case the improved test accuracy could not have come from pure memorization because 100% of the training examples are incorrectly labeled. However, since in each wrong answer of a N-people puzzle, there are still $N - \tilde{N}$ correct role assignments where the random $\tilde{N} \geq 1$. The model might have learned to reason from those partially correct role assignments in the wrong answer.

However, as shown in Fig. 33, when applied to more capable model GPT4o-mini, Direct FT on 5-people puzzles where 100% training examples have corrupted answers does not lead to improvement. Moreover, the negative effects transfer to other tasks, notably easier ones (2/3/4-ppl). Nevertheless, as the percentage of corrupt-answer training examples reduces ($\leq 50\%$), the model could gain improved and generalizable reasoning capabilities.

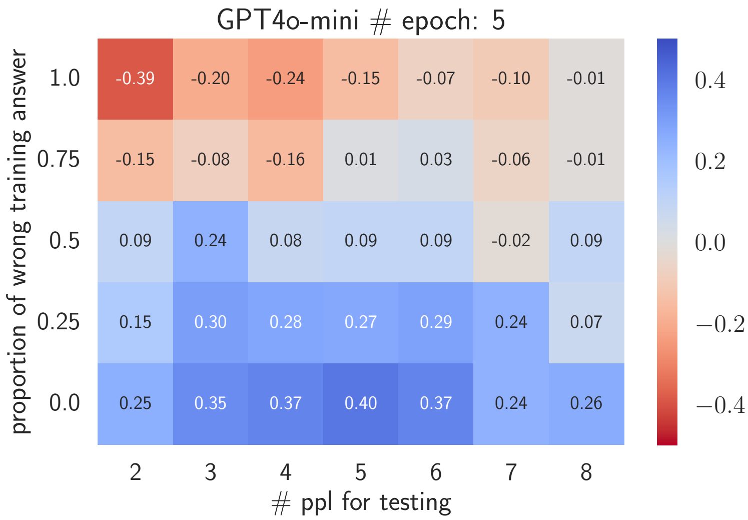
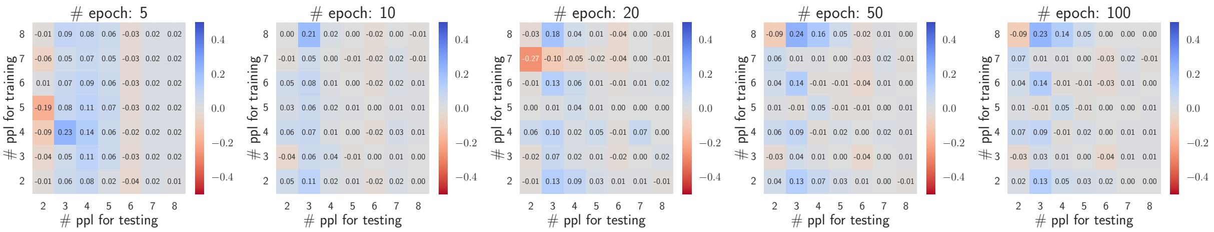
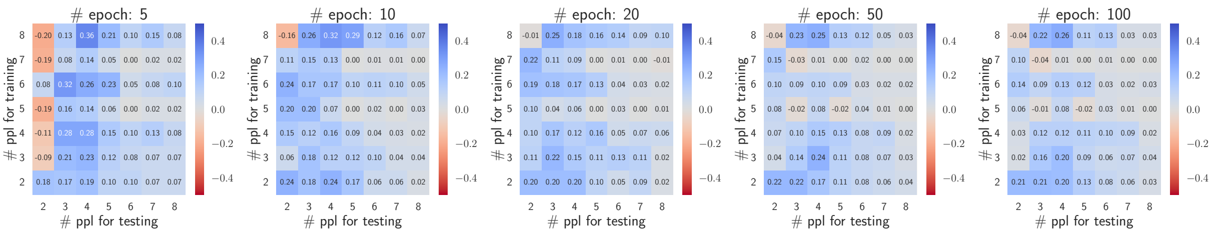


Figure 33: Direct FT w/ various wrong training answer % on 5-ppl task.

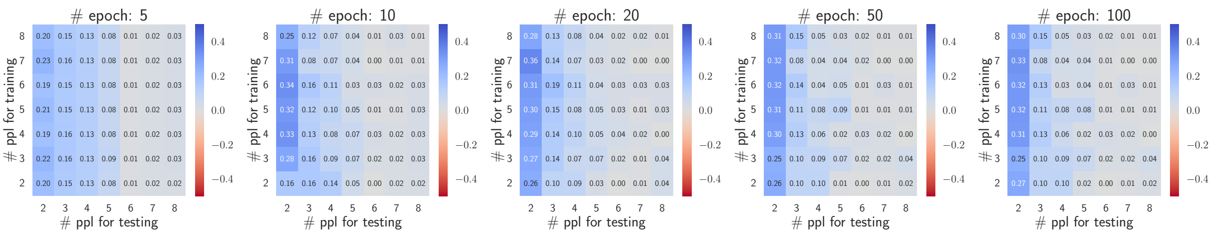
Fig. 37 displays the results of direct fine-tuning using 5-people training K&K puzzles for the GPT4o-mini model, containing varying percentages of incorrect answers in the dataset: $100\%, 75\%, 50\%, 25\%, 0\%$. This is evaluated across different epochs in the five-person puzzle. As noted in § E.4, when the training dataset includes 50% or fewer samples with incorrect answers, fine-tuning can still enhance K&K's performance across various testing tasks.



(a) 0-shot Direct Prompting

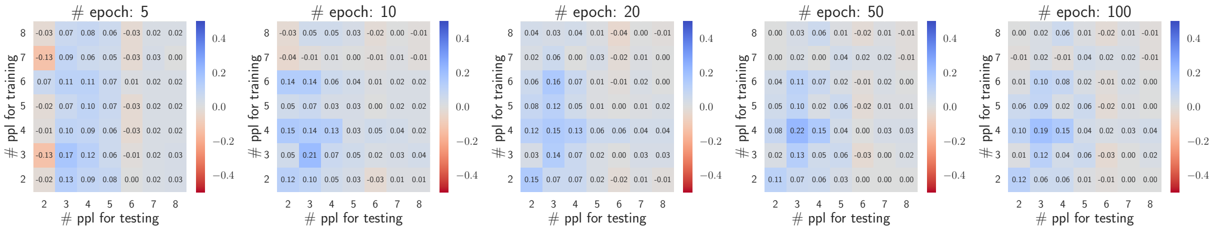


(b) 0-shot CoT Prompting

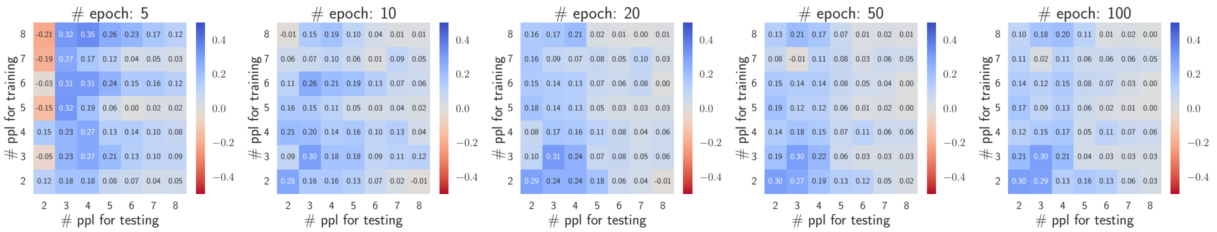


(c) 1-shot Direct Prompting

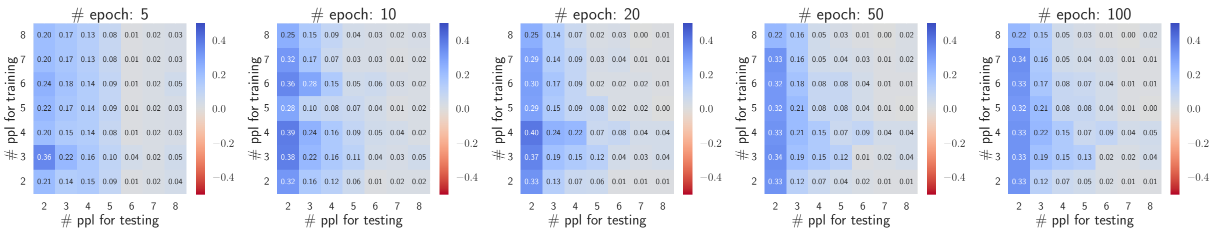
Figure 34: Improvement in test accuracy on N-person problems for Llama3-8B Direct FTed on M-person problems **with completely wrong answers**, compared to the unfine-tuned model, under various evaluation configurations.



(a) 0-shot Direct Prompting

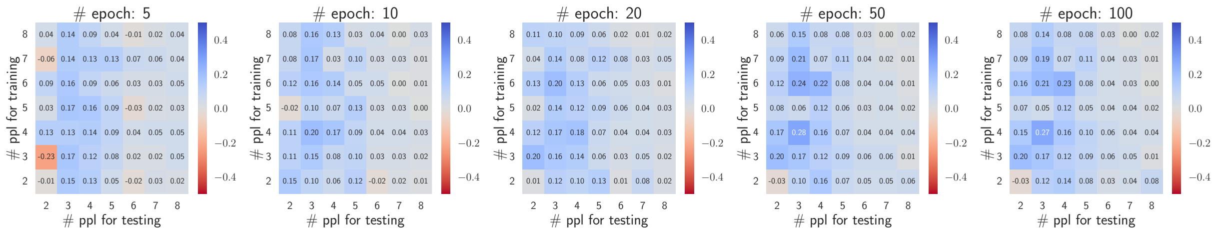


(b) 0-shot CoT Prompting

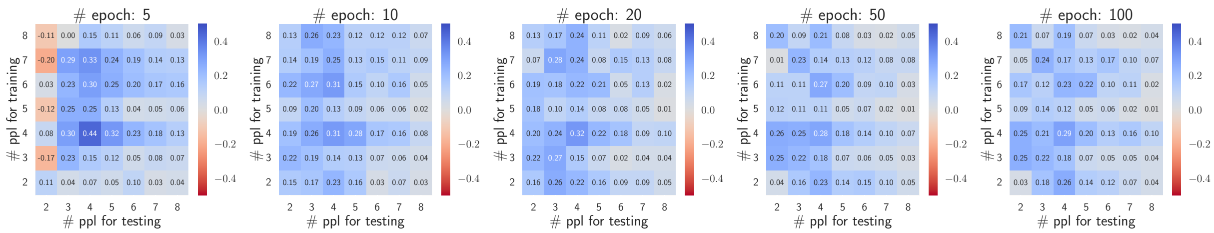


(c) 1-shot Direct Prompting

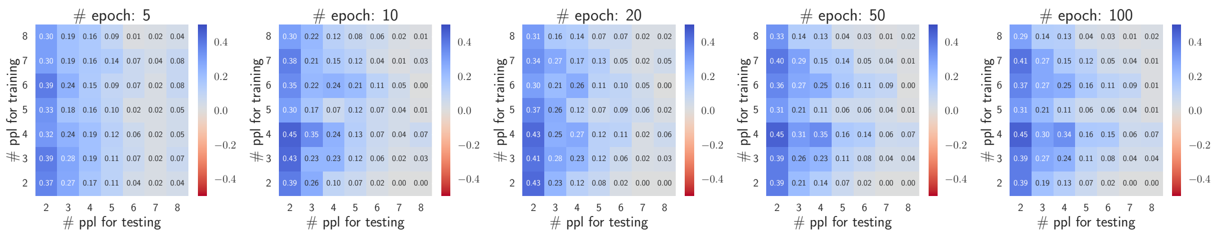
Figure 35: Improvement in test accuracy on N-person problems for Llama3-8B Direct FTed on M-person problems **with 75% wrong answers**, compared to the unfine-tuned model, under various evaluation configurations.



(a) 0-shot Direct Prompting



(b) 0-shot CoT Prompting



(c) 1-shot Direct Prompting

Figure 36: Improvement in test accuracy on N-person problems for Llama3-8B Direct FTed on M-person problems **with 50% wrong answers**, compared to the unfine-tuned model, under various evaluation configurations.

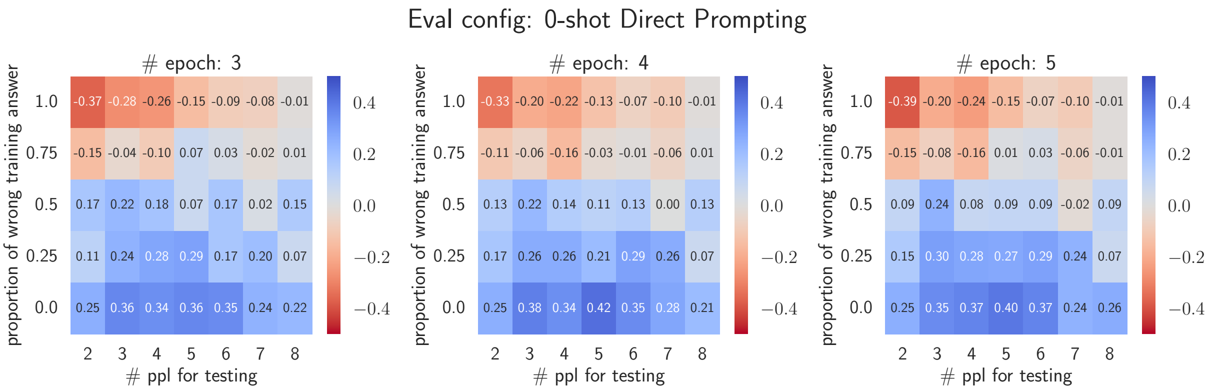


Figure 37: Improvement in test accuracy on N-people problems for GPT4o-mini fine-tuned on 5-people problems with different proportion of wrong answers, compared to the unfine-tuned model. Direct FT with 50% wrong answers still improves K&K performance.

E.5 Probing

We report the probing accuracy for the un-fine-tuned Llama3-8B model in Fig. 38. As shown, without fine-tuning, the model demonstrates relatively low probing accuracy, with values usually below 90%.

Figure 38: Probing accuracy of K&K puzzles with different number of people in testing puzzles across different layers of the un-finetuned Llama3-8B transformer model.

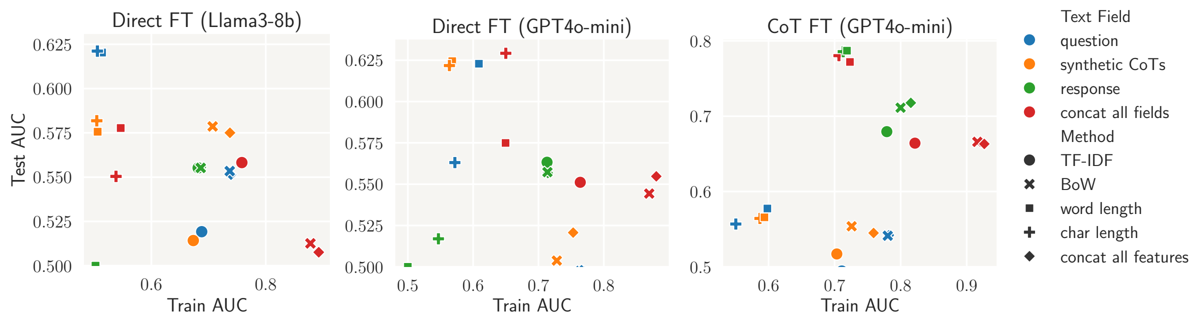


Figure 39: AUC for classifying 3-people puzzles under leaf perturbation based on puzzle-based indicators. Results under more tasks and perturbations are in Fig. 40.

E.6 Distinguishing Memorization from Reasoning

Setup. We collect training samples where the targeted LLM predicts correctly, and label each as either "consistently solved" (reasoning) or "not consistently solved" (memorization). We then train a logistic regression classifier using an 8:2 train/test split to distinguish between puzzles the model solves through reasoning versus memorization.

Puzzle-based indicators. We consider the following features: (1) TF-IDF; (2) Bag-of-Words; (3) Word Length; (4) Character Length; (5) concatenation of all. Each feature can be extracted from one of the following fields: (1) question; (2) synthetic CoT reasoning steps; (3) model response[6]; (4) concatenation of the above fields. The train and test performance (measured with AUC as the dataset can be unbalanced) are shown in Fig. 39. We observe a test AUC of 0.629/0.787 for Direct/CoT FT-ed GPT4o-mini, and 0.627 for Direct FT-ed Llama3-8B. This indicates that the puzzle-based indicators could be informative, though not perfect, at determining which examples are reasoned vs. memorized.

Fig. 40 shows the train and test AUC for predicting whether N-person puzzles can be consistently solved by a specific model under perturbations, using puzzle-based indicators. The results indicate that length-related features are useful for distinguishing memorization from reasoning. Notably, the test AUC is generally higher for CoT FTed GPT4o-mini compared to Direct FTed GPT4o-mini.

Model-based indicators. We also study model-based indicators to test whether the internal activations of the fine-tuned model are informative for this categorization. Since we cannot access model internals of GPT4o-mini, we conduct the experiment on Llama3-8B. Specifically, we feed each puzzle question into the fine-tuned model, extract average embeddings from each layer, and train a linear classifier per layer. Appendix Fig. 41 shows test AUCs, where we also compare the fine-tuned model to its non-fine-tuned counterpart. We observe that **(1)** Lower-layer features poorly distinguish memorization from reasoning, but higher layers improve. **(2)** The features from the FTed model are consistently more informative than the un-FTed one, suggesting that the model's decision regarding memorization vs. reasoning on specific samples likely stems from the fine-tuning process. **(3)** The best embedding-based indicator (0.70 AUC) outperforms the puzzle-based indicator (Fig. 39 left, 0.627 AUC) on 3-people puzzles.

We report test AUC for classifying puzzles based on whether they are consistently solved under leaf/statement perturbation by the Llama3-8B model Direct-FTed on the 3/5-person task. As shown in Fig. 41 and Fig. 42, the embeddings across different layers of the fine-tuned Llama3-8B provide more distinguishable signals for memorized samples than those of the base model.

[6]Strictly speaking, this is a model-based indicator feature.

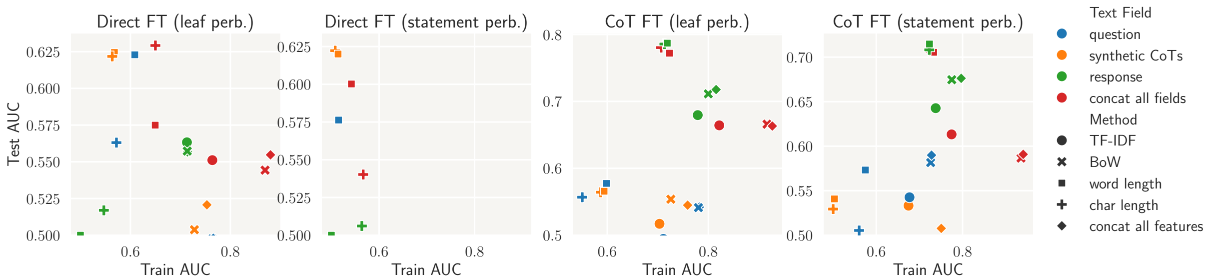
E.7 DynamicZebra

Here we fine-tune Qwen2.5-7B-Instruct-1M on the question-answer pairs (Direct FT) using LoRA with a learning rate of 5e-5 and batch size of 32.

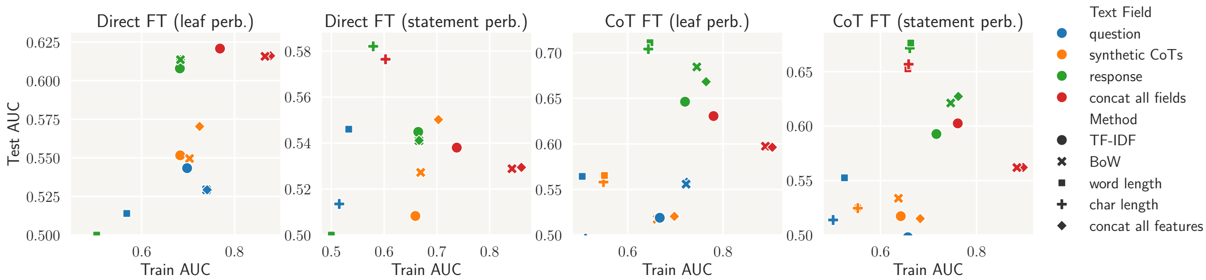
In Fig. 11, the accuracy and memorization score on DynamicZebra perturbed training set (we report results using the checkpoint at 3510 steps) shows that the model still relies on training examples and cannot effectively solve perturbed training variants.

In Fig. 44, the accuracy on the DynamicZebra test set (full puzzle accuracy and cell accuracy of the puzzle grid) shows that Direct FT-ed models can generalize to solve unseen puzzles at different difficulty levels.

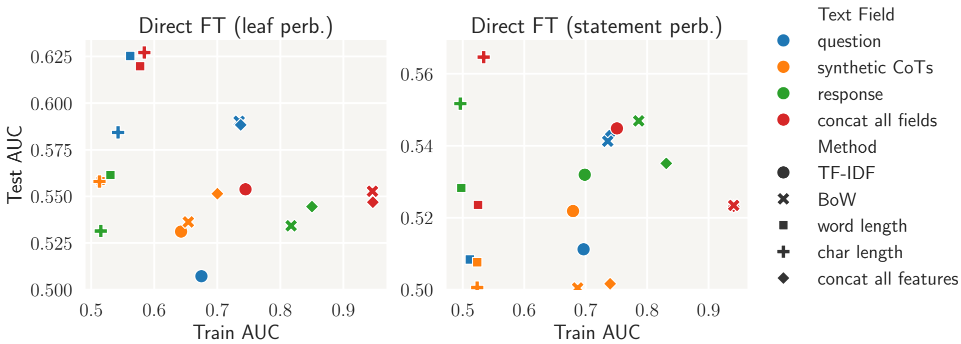
These results demonstrate the effectiveness of our framework in measuring reasoning and memorization on structured logical reasoning tasks. We believe DynamicZebra complements our findings on the K&K dataset by validating the methodology in a different logic domain with a completely different problem format.



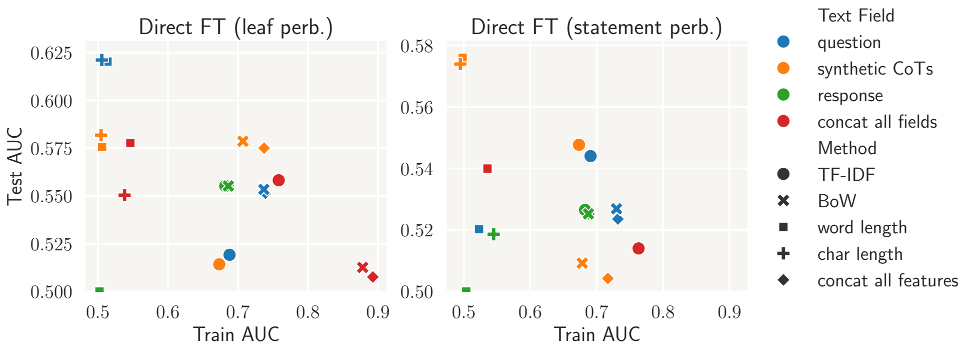
(a) 3-person puzzles for GPT4o-mini.



(b)5-person puzzles for GPT4o-mini.



(c) 3-person puzzles for Llama3-8B.



(d) 5-person puzzles for Llama3-8B.

Figure 40: AUC for predicting whether N-person puzzles can be consistently solved under perturbations based on puzzle-based indicators.

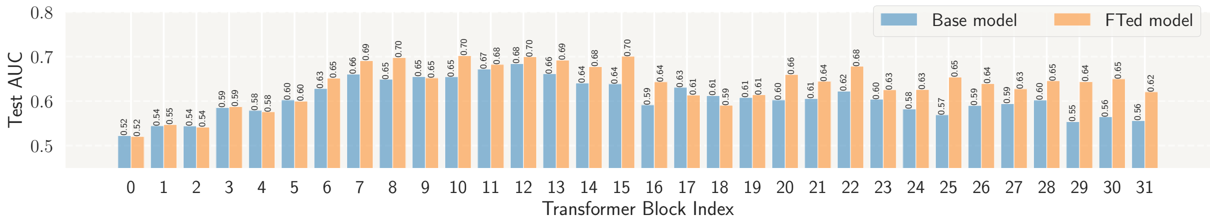
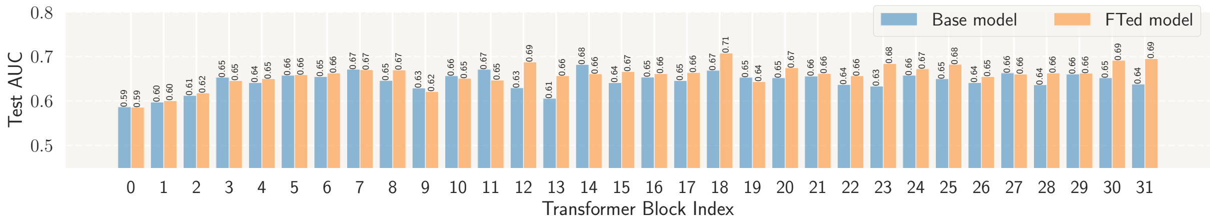
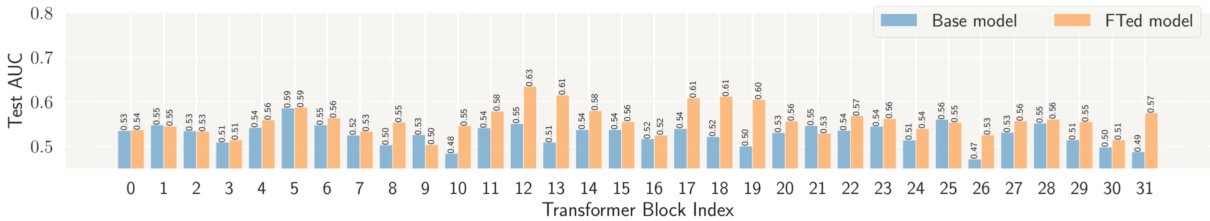


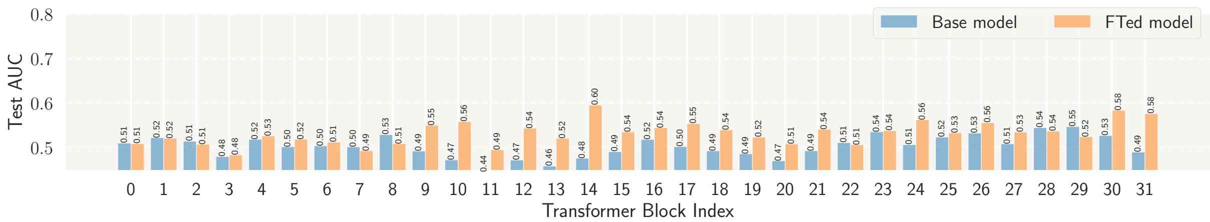
Figure 41: Test AUC for predicting 3-people puzzles based on whether they are consistently solved under leaf perturbation by the Llama3-8B model Direct-FTed. The embeddings across different layers of the fine-tuned Llama3-8B provide more distinguishable signals than those of the un-FTed model, leading to 0.7 AUC at the middle layers. Results under more tasks and perturbations are in Fig. 42.



(a) 3-person puzzles under statement perturbation.



(b) 5-person puzzles under leaf perturbation.



(c) 5-person puzzles under statement perturbation.

Figure 42: Test AUC for predicting whether N-person puzzles can be consistently solved under perturbations by Direct-FTed Llama3-8B models.

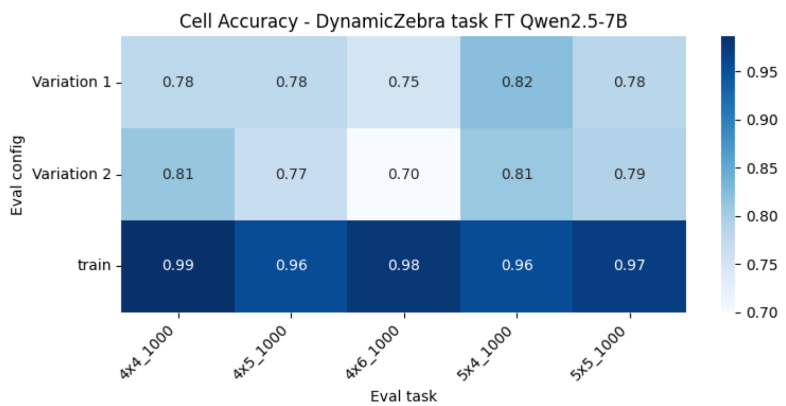
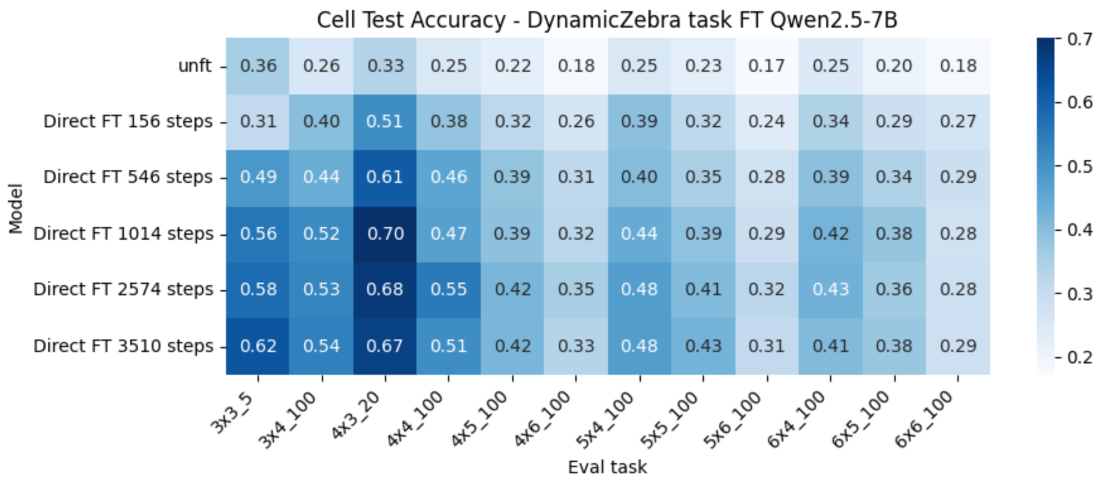
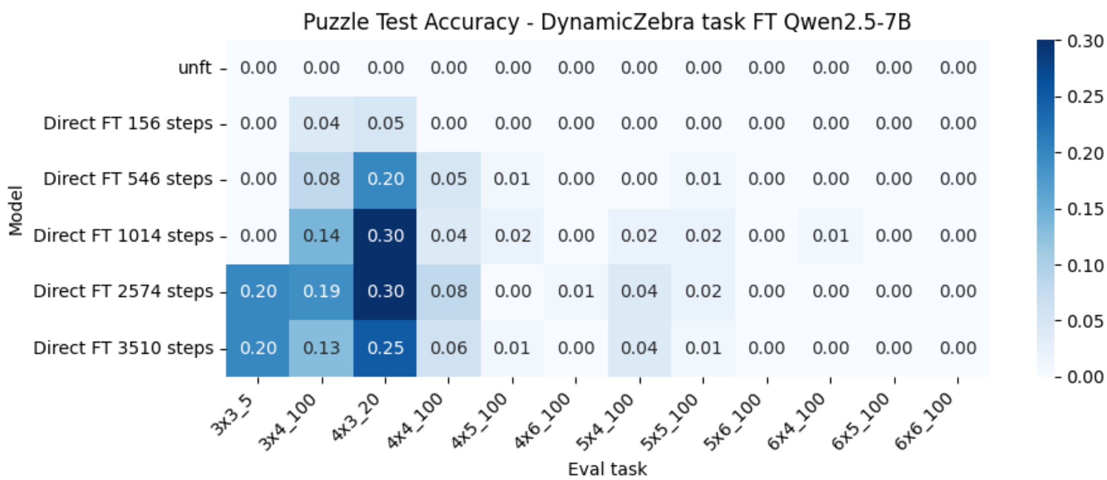


Figure 43: Cell level accuracy on training set.



(a) Cell-level accuracy on test set.



(b) Puzzle-level accuracy on test set.

Figure 44: DynamicZebra test accuracy for Direct-FTed Qwen2.5-7B.