FINEREASON: Evaluating and Improving LLMs' Deliberate Reasoning through Reflective Puzzle Solving

Anonymous ACL submission

Abstract

Many challenging reasoning tasks require not just rapid, intuitive responses, but a more deliberate, multi-step approach. Recent progress in large language models (LLMs) highlights an important shift from the "System 1" way of quick reactions to the "System 2" style of reflection-and-correction problem solving. However, current benchmarks heavily rely on the final-answer accuracy, leaving much of a model's intermediate reasoning steps unexamined. This fails to assess the model's ability to reflect and rectify mistakes within the reasoning process. To bridge this gap, we introduce FINEREASON, a logic-puzzle benchmark for fine-grained evaluation of LLMs' reasoning capabilities. Each puzzle can be decomposed into atomic steps, making it ideal for rigorous validation of intermediate correctness. Building on this, we introduce two tasks: **state checking**, and **state transition**, for a comprehensive evaluation of how models assess the current situation and plan the next move. To support broader research, we also provide a puzzle training set aimed at enhancing performance on general mathematical tasks. We show that models trained on our state checking and transition data demonstrate gains in math reasoning from 82.3% to 87.4% on GSM8K using the DeepSeek-R1-Distill-Qwen-7B model.

1 Introduction

In cognitive science, human reasoning is typically characterized by two distinct systems: (i) System 1, which is fast, automatic, and effortless, and (ii) System 2, which is slow, analytical, and effortful (Kahneman, 2011). System 2 reasoning enables humans to proactively anticipate future outcomes, reassess intermediate states, and iteratively refine solutions (Yao et al., 2023), thereby allowing them to tackle more complex reasoning tasks. Recent studies (OpenAI, 2024; Snell et al., 2024; Guo et al., 2025; Team, 2025) suggest that large language models (LLMs) can attain advantages akin to those of System 2 reasoning. Instead of generating answers directly, these models can engage in iterative reflection and correction to refine their reasoning processes (Shinn et al., 2023), leading to impressive performance on reasoning-intensive tasks such as mathematics and coding (Qin et al., 2024; Muennighoff et al., 2025).

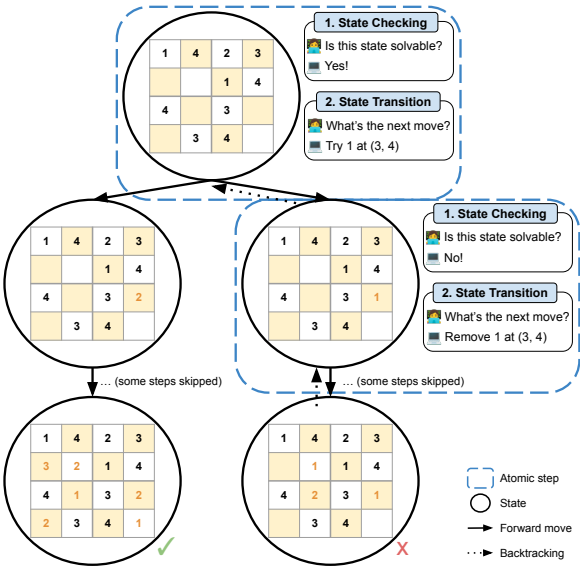


Figure 1: An illustration of stepwise state checking and transition in a Sudoku solution tree.

Despite these improvements, the underlying reasoning mechanisms responsible for these enhancements remain underexplored. Existing reasoning benchmarks primarily focus on the final-answer accuracy (Hendrycks et al., 2020; Cobbe et al., 2021; Chen et al., 2021; Hendrycks et al., 2021), which offers limited insight into LLMs' internal reasoning processes, such as reflection and correction. For instance, a model might reach a correct conclusion through flawed reasoning (Zelikman et al., 2022; Creswell et al., 2023; Lightman et al., 2024). This diminishes the trustworthiness of model's outputs, posing potential risks in practical usages. Moreover, models may achieve high accuracy by exploit-

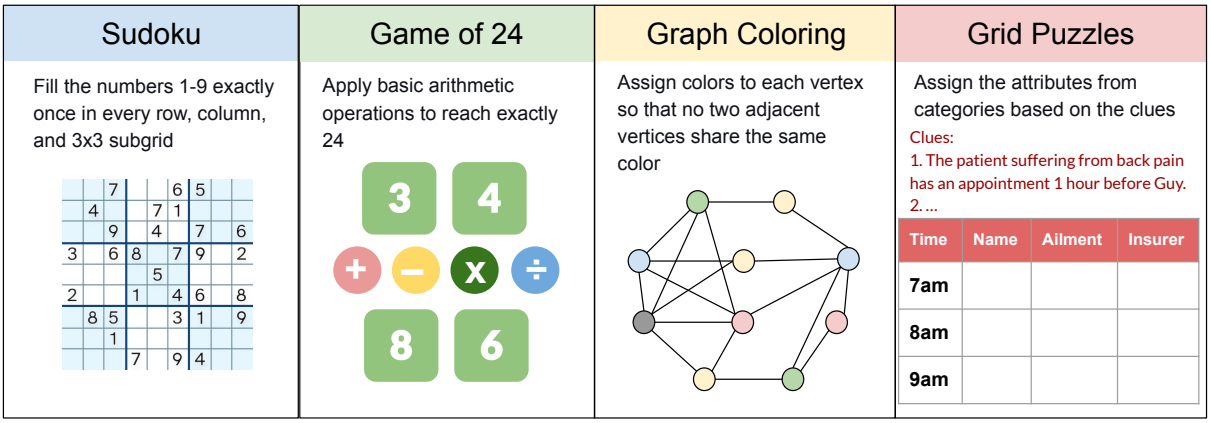


Figure 2: An illustration of four puzzle categories in FINEREASON.

ing superficial patterns in the training data (Roelofs et al., 2019; Enström et al., 2024), making it difficult to ascertain whether the observed performance gain truly stems from deliberative reasoning. Therefore, there is a pressing need for more comprehensive reasoning benchmarks that assess the integrity of intermediate processes.

In this work, we present FINEREASON, a logic-puzzle benchmark for granular evaluation of LLMs' reasoning capabilities. FINEREASON includes four types of puzzles: *Sudoku*, *Graph Coloring*, *Game of 24*, and *Grid Puzzles*. Solving a logic puzzle involves a series of discrete steps, and its explicit rules make it straightforward for validating the intermediate states. To structure our evaluation, we propose two key actions: **state checking** and **state transition** in each atomic step, as shown in Figure 1. State checking predicts whether the current state can lead to a solvable solution (Agarwal et al., 2019; Wang et al., 2024). It captures both retrospective evaluation of prior steps, a common reasoning pattern in current LLMs' outputs (Lightman et al., 2024), and prospective analysis of future steps. Meanwhile, state transition focuses on determining the next valid step, either moving forward or backtracking to the previous state. Together, these two tasks cover the entire puzzle-solving process, revealing the internal reasoning processes of reflection, correction, and exploration of alternative paths in LLMs.

Our evaluation results show that OpenAI-o1 (OpenAI, 2024) outperforms Gemini-2.0-Flash-Thinking (Google, 2024) by a large margin of 19.7%, despite both being reasoning-oriented models that already excel at common math and coding tasks. On the other hand, general-purpose models, such as GPT-4o (OpenAI et al., 2024), GPT-3.5 (OpenAI, 2022), Qwen2.5-72b-Instruct (Qwen et al., 2025), and Gemini-2.0-Flash (Google, 2024) struggle with deep reasoning, often making near-random guesses during state checking and showing extremely low performance in state transition task. To improve models' reasoning capabilities, we develop a specialized training set about state checking and transition. Integrating our dataset with math training data consistently boosts performance on mathematical reasoning. For example, when applied to GSM8K using the DeepSeek-R1-Distill-Qwen-7B model, our state checking and transition data improve overall math reasoning accuracy from 82.3% to 87.4%, compared to models trained exclusively on math data. This suggests that our data offers a straightforward way to boost the reasoning abilities of LLMs, thereby enhancing performance on other reasoning-intensive tasks.

Our main contributions are three-fold: 1) We introduce FINEREASON, a fine-grained, puzzle-based benchmark accompanied by systematic evaluations on state checking and transition, to comprehensively evaluate reasoning capabilities of LLMs. 2) Our experiments reveal substantial limitations in deep reasoning among general-purpose models, and even in the leading reasoning model, Gemini-2.0-Flash-Thinking, leaving substantial room for improvement. 3) We develop a supplementary training set focused on puzzles, which enhances general mathematical reasoning in LLMs.

2 FINEREASON

We present FINEREASON, a logic-puzzle benchmark that comprehensively assesses LLMs' reasoning abilities through stepwise evaluation of state checking and transition. An overview of each puzzle is shown in Figure 2. Inspired by the adage

Puzzle	Puzzle State	Minimal Move
Sudoku	Partial / Complete 9x9 board	Add / Remove a digit
Graph Coloring	A graph of partially / completely colored vertex	Color / Uncolor a vertex
Game of 24	Partial / Complete arithmetic expression	Apply / Unapply an operation to two remaining numbers
Logic Grid Puzzles	Partial / Complete grid	Assign / Remove attributes according to a given clue

Table 1: The definition of minimal move for each category of logic puzzles in our FINEREASON.

"think twice before acting," the two actions capture how models assess the current situation (*i.e.,* state checking) and plan the next move (*i.e.,* state transition), which is essential for deliberate reasoning.

Formally, we represent the reasoning process of a logic puzzle as $p = \{p_1, p_2, ..., p_n\}$, where n denotes the number of atomic steps. Each step p_i consists of a puzzle state s_i and two actions: state checking a_i^c and state transition a_i^t, *i.e.,* $p_i = (s_i, a_i^c, a_i^t)$. Applying these actions to s_i produces the next state s_{i+1}. The puzzle-solving process begins at an initial state s_1 and proceeds through a sequence of such atomic steps until reaching the solution state s_n that satisfies all constraints.

In the following section, we first introduce a tree-based approach for step decomposition(§2.1) in our puzzles. We then describe the two key actions – state checking and state transition – that facilitate fine-grained reasoning evaluation of models (§2.2).

2.1 Tree-based Puzzle Decomposition

Puzzle solving can be represented as a tree, where nodes correspond to intermediate states describing the completion stage of the current puzzle, and edges represent the execution of state checking and state transition, as illustrated in Figure 1. Edges are bidirectional, enabling both the exploration of child states and backtracking to the parent state when encountering dead ends. This process begins at the root node s_1 and terminates at a solution leaf s_n, potentially requiring multiple backtracks to explore alternate paths.

To capture all possible states, we perform a depth-first search (DFS) from the initial puzzle state s_1 until no further valid states remain for exploration. Each DFS step involves a minimal move to ensure that no valid state is overlooked. For example, in Sudoku, we add or remove only a single digit at each step. Table 1 summarizes the definition of a minimal move for each puzzle category. Additionally, we translate puzzle rules into executable code to automatically validate each state.

Sudoku. Given a partially filled 9×9 Sudoku grid, the aim is to fill the remaining cells such that each row, each column, and each of the nine 3×3 subgrids contains all digits from 1 to 9 exactly once. A Sudoku state can be either a partially or fully completed 9×9 grid. The former refers to any intermediate state s_i encountered during the reasoning process, while the latter is the final solvable state s_n. A minimal move consists of either adding a number to the grid for exploration or removing a number for backtracking. Our Sudoku questions are collected from Kaggle.[1]

Graph Coloring. The puzzle of graph coloring is to assign colors to each vertex in a graph such that no two adjacent vertices share the same color. Each puzzle specifies a maximum number of colors allowed in a graph. A graph coloring state is either a partially colored graph, denoted as s_i, or a completely colored graph, denoted as s_n. A minimal move involves either assigning a color to a vertex or removing a color from a vertex. To create the questions, we generate random graphs and find their respective chromatic numbers using the backtracking algorithm (van Beek, 2006).

Game of 24. In Game of 24, given four numbers, the task is to apply basic arithmetic operations (addition, subtraction, multiplication, and division) in any order to reach exactly the value of 24. Each number must be used exactly once. Each state is a partial or complete arithmetic expression. The minimal move is to apply or unapply an operation to two of the remaining numbers. We use the dataset from Yao et al. (2023).

Logic Grid Puzzles. Logic grid puzzles require filling a grid with attributes from multiple categories (*e.g.,* color, time) based on a set of textual clues. Each attribute should appear exactly once per category and satisfy all given clues. Each state can be a partially filled or fully completed grid, with the minimal move being adding or removing a combination of attributes specified in a clue. Unlike previous puzzles, translating textual clues into verification code is challenging in logic grid puzzles, especially when it involves attribute compar-

[1] https://www.kaggle.com/datasets/bryanpark/sudoku

isons. To address this issue, we define three functions: $r(v)$ and $c(v)$ to retrieve the row and column numbers of an attribute v, and $T(i, j)$ to identify the attribute at row i and column j. These functions encode attributes to a structured axis space. Thus, the textual clues can be parsed into conditions involving $r(v)$, $c(v)$, and $T(i, j)$ for constraint checks. For example, Clue 1 in Figure 2 can be parsed to $T(r(\text{“Guy”}), c(\text{“Time”})) - T(r(\text{“back pain”}), c(\text{“Time”})) == 1$. We annotate one sample for one-shot prompting using GPT-4o for initial parsing, followed by manual verification. We ensure all solutions pass the coded clues. Our grid puzzle data is constructed from the dataset collected by Tyagi et al. (2024).

2.2 Evaluation Tasks

We define two key actions, *state checking* and *state transition*, which enable movement between states.

State Checking. State checking involves assessing if a given state s_i can lead to a solvable leaf s_n. Based on our constructed puzzle tree, we label a state as solvable if at least one valid solution exists in the subtree of s_i. To ensure a diverse difficulty range, we uniformly sample both solvable and unsolvable states across different tree depths. For each sampled state, we prompt models to evaluate its solvability with puzzle rules, prior visited states, and the current state (see Appendix A.2). In general, state checking evaluates two key aspects: 1) It tests if a model can reflect on prior steps (*i.e.,* states and actions) to avoid rule violations (e.g., preventing duplicate "1"s in a Sudoku row). 2) It requires models to anticipate potential dead ends by looking ahead. The second aspect, however, requires a higher level of reasoning to proactively detect states that are unsolvable.

State Transition. State transition involves making the minimal move defined in §2.1, which requires models to determine the next valid state. Based on the state-checking results, models should proceed if the state is solvable and backtrack otherwise. Specifically, at a solvable state, a correct transition would be an unvisited child of the given state. At an unsolvable state, the correct move is to backtrack to its parent state. To isolate the impact of state transition from incorrect state checking, our evaluation provides the ground-truth state-checking labels. We sample states from the puzzle tree and construct prompts with puzzle rules, prior visited states, the sampled state, and some unsolvable child states (see Appendix A.2). The inclusion of unsolvable child states is to assess whether models can effectively bypass these bad states.

Puzzle	Model	End-to-end Acc.
Sudoku	GPT-4o	0
	GPT-3.5	0
	Gemini-F	**5.9**
	Qwen2.5-Inst	0
	Gemini-FT	0
	o1	0
Graph Coloring	GPT-4o	7.8
	GPT-3.5	3.9
	Gemini-F	35.3
	Qwen2.5-Inst	2.0
	Gemini-FT	**80.4**
	o1	78.4
Game of 24	GPT-4o	15.3
	GPT-3.5	3.1
	Gemini-F	**83.7**
	Qwen2.5-Inst	17.3
	Gemini-FT	48.0
	o1	54.1
Grid Puzzles	GPT-4o	2.2
	GPT-3.5	2.2
	Gemini-F	10.9
	Qwen2.5-Inst	4.4
	Gemini-FT	34.8
	o1	**45.7**

Table 2: A preliminary study of End-to-end puzzle-solving performance of LLMs.

3 Experimental Setup

Datasets. We sample 500 intermediate states per puzzle category, resulting in 2000 test instances for each task: state checking and state transition. Dataset statistics are included in Appendix A.1.

Implementations. To elicit the reasoning capability of LLMs, we leverage the zero-shot chain of thought (CoT) prompting (Kojima et al., 2022). To ensure a genuine evaluation of LLM's inherent reasoning capabilities, we explicitly include the instruction "`Do not solve using programming`" in the prompt, restricting the models from relying on programmatic solutions.

Models. We select the best-performing open and closed-source models, including 1) reasoning-oriented models with deep thinking: o1 (OpenAI, 2024), Gemini-2.0-Flash-Thinking (Gemini-FT, Google (2024)), and 2) general-purposed models that perform straightforward execution: GPT-4o (OpenAI et al., 2024), GPT-3.5 (OpenAI, 2022), Gemini-2.0-Flash (Gemini-F, Google (2024)), and

Puzzle	Model	SC Acc.	ST Acc.	Avg.
Sudoku	Random	50.0	-	-
	GPT-4o	52.4	38.8	45.6
	GPT-3.5	49.0	10.6	29.8
	Gemini-F	50.4	39.0	44.7
	Qwen2.5-Inst	51.6	26.6	39.1
	Gemini-FT	69.2	48.8	59.0
	o1	**81.0**	**70.2**	**75.6**
Graph Coloring	Random	50.0	-	-
	GPT-4o	56.4	49.4	52.9
	GPT-3.5	52.2	20.4	36.3
	Gemini-F	56.8	45.8	51.3
	Qwen2.5-Inst	58.6	35.4	47.0
	Gemini-FT	92.6	46.4	69.5
	o1	**94.6**	**65.0**	**79.8**
Game of 24	Random	50.0	-	-
	GPT-4o	82.6	23.0	52.8
	GPT-3.5	56.4	14.2	35.3
	Gemini-F	93.4	54.6	74.0
	Qwen2.5-Inst	88.2	39.2	63.7
	Gemini-FT	96.0	48.6	72.3
	o1	**97.4**	**86.6**	**92.0**
Grid Puzzles	Random	50.0	-	-
	GPT-4o	52.4	10.0	31.2
	GPT-3.5	42.6	11.4	27.0
	Gemini-F	37.4	18.8	28.1
	Qwen2.5-Inst	40.8	9.6	25.2
	Gemini-FT	**89.0**	51.4	70.2
	o1	88.8	**77.6**	**83.2**

Table 3: The state checking and transition accuracy using FINEREASON, where **SC** and **ST** denote state checking and transition, respectively.

Qwen2.5-72B-Instruct (Qwen2.5-Inst, Qwen et al. (2025)).

4 Evaluation Results

4.1 Preliminary: End-to-End Puzzle Solving

We report a preliminary evaluation of end-to-end puzzle solving in Table 2. Despite their exceptional performance in coding and math (Qwen et al., 2025), these models perform notably weak on end-to-end puzzle solving. Additionally, there are some counter-intuitive observations. Gemini-F outperforms Gemini-FT on Sudoku and Game of 24 but is less effective on the other two puzzles. These inconsistencies raise questions about the reliability of using end-to-end puzzle solving as a metric for evaluating LLMs' reasoning capabilities.

4.2 Main results

Table 3 summarizes the overall results of state checking and transition across different puzzles. The results reveal noticeable performance gaps between reasoning-oriented and general-purposed models on both tasks. For the state-checking task, reasoning-oriented models, like o1 and Gemini-FT, lead the performance and are consistently superior to the random baseline in every puzzle. In contrast, general-purposed models barely reach or slightly surpass the random baseline in tasks such as Sudoku and Graph Coloring. A similar trend is held in the state-transition task, especially in the Game of 24 and Grid Puzzles. These findings verify that inference-time scaling can significantly enhance reasoning capabilities (Snell et al., 2024; Muennighoff et al., 2025).

For reasoning models, Gemini-FT is on par with o1 in state checking but consistently lags behind o1 in state transition across all puzzle categories. This indicates, compared to o1, Gemini-FT has certain shortcomings in its reasoning process, particularly in error revision. These findings align well with our practical experience of using these models, which provides empirical evidence that FINEREASON reflects a more accurate evaluation of LLMs' reasoning capabilities.

Moreover, we observe that most models exhibit a significant "execution gap": they perform better in state checking than in state transition. This suggests that instead of correctly progressing through intermediate states, models tend to take shortcuts to reach the final answer.

4.3 State-Checking Patterns

A key feature of state checking is to lookahead at potential paths. To analyze models' behaviorals in this task, we report the state-checking precision, recall, and F1 scores in Table 4. We designate unsolvable states as positive cases. Therefore, recall measures how likely a model detects the dead ends, whereas precision shows the reliability of predicting unsolvable states.

We find that reasoning models can reliably draw state-checking conclusions. For general models (GPT-4o, Gemini-F, Qwen2.5-Inst), we observe contrasting behaviors in different tasks. In Sudoku, the recall is generally low. However, they can effectively detect dead ends in the game of 24, as indicated by the high recall scores. This difference could be attributed to a deeper puzzle tree of Sudoku, posing a greater challenge in detecting unsolvable states. The above results suggest that general LLMs tend to make overly optimistic decisions (i.e., predicting solvable) when faced with tasks that exceed their actual capabilities.

Nevertheless, GPT-4o and Qwen2.5-Inst show high precision. This suggests that these models are conservative and might not attempt to classify states as unsolvable unless they are very confident.

We have also observed significant gaps between

Puzzle	Model	Recall	Precision	F1
Sudoku	GPT-4o	6.4	80.0	11.9
	GPT-3.5	28.0	49.0	35.6
	Gemini-F	3.2	57.1	6.06
	Qwen2.5-Inst	4.8	75.0	9.02
	Gemini-FT	**87.2**	64.3	74.0
	o1	73.2	**86.7**	**79.4**
Game 24	GPT-4o	95.6	75.9	84.6
	GPT-3.5	54.8	56.6	55.7
	Gemini-F	98.8	89.2	93.7
	Qwen2.5-Inst	97.6	82.2	89.2
	Gemini-FT	94.8	97.5	96.1
	o1	**95.6**	**99.2**	**97.4**

Table 4: Precision, recall, and F1 scores of state-checking task in FINEREASON.

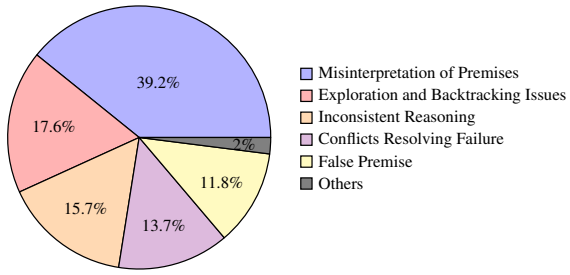


Figure 3: Human analysis of error types.

recall and precision in the other tasks: Graph Coloring and Grid Puzzles as reported Appendix A.4.

4.4 Quality Analysis of State Checking

To examine the errors made in state checking, we conduct a human analysis of the mistakes from the best-performing model, o1, in Figure 3. The most common error is Misinterpretation of Premises, where o1 incorrectly uses available information to reach a faulty conclusion. For instance, in a grid puzzle, given the clue "The chocolate piece, Joey's cake, and the $125 cake are three different cakes," o1 may still mistakenly consider Joey's cake and the $125 cake to be the same. Additionally, the model sometimes fails to explore alternative paths, leading to incorrect classification. Other mistakes include drawing a wrong conclusion despite correct deductions (Inconsistent Reasoning), failure to recognize conflicting information (Conflicts Resolving Failure), nonexistent constraints (False Premise), and a few instruction-following errors.

4.5 State-Transition Performance Breakdown

The left part of Figure 4 shows the Sudoku state-transition performance breakdown for each class (solvable vs. unsolvable). We can observe that most models transit much better on solvable states than on unsolvable ones. The large gap indicates that models are better at proceeding forward from a valid state than backtracking. This finding may contribute to a forward-generation reasoning style of LLMs. This trend does not apply to GPT-3.5, which shows weak performance and tends to rely primarily on random guessing.

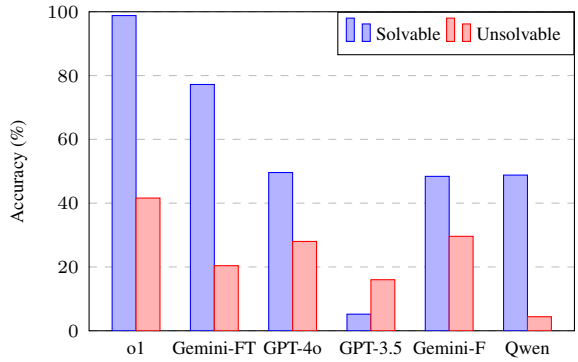
The right chart illustrates the errors typically made by each model. Starting from solvable states, there are three common errors: making multiple moves (Multiple Moves), the move violating Sudoku rules (Invalid Move), transitioning to an unsolvable child state (Unsolvable Child) In contrast, when transitioning from unsolvable states, two primary errors are: failing to return to the parent state (Backtracking failure) and making an additional move to a sibling state after backtracking (Sibling). Among these errors, Backtracking Failure is the most frequent across all models. Models sometimes jump back more than one level (*e.g.,* to a grandparent state) or to a wrong parent state, indicating that LLMs struggle with the step-by-step backtracking. For reasoning models (o1 and Gemini-FT), sibling is the second most frequent error. This error arises because the models violate the minimal move principle (Table 1), highlighting a deficiency in their instruction-following capability. For general models, they also frequently commit an invalid move. This shows that general models often fail to adequately check Sudoku rules.

4.6 Performance by Difficulty Level

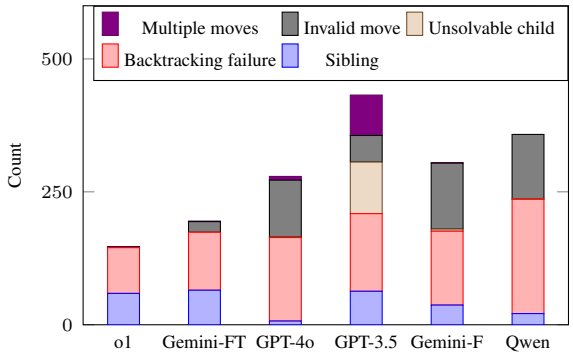
To understand the state-checking performance across difficulty levels, we plot the density diagrams based on the number of unfilled cells in the current Sudoku state. Sudoku states with fewer empty cells are more likely to be predicted correctly. As the number of unfilled cells increases, the problem becomes more complex and requires more exploration, the proportion of incorrect predictions increases.

5 Training Results

As highlighted in §4, most models still lack state-checking and state-transition abilities, which are essential for reasoning. We hypothesize that training on these tasks can enhance performance on other reasoning-intensive tasks like math. To validate our hypothesis, we construct a training set consisting of state-checking and state-transition data from Su-



(a) State-transition performance breakdown by class



(b) State-transition error type breakdown

Figure 4: Performance breakdown and error analysis of state-transition in Sudoku.

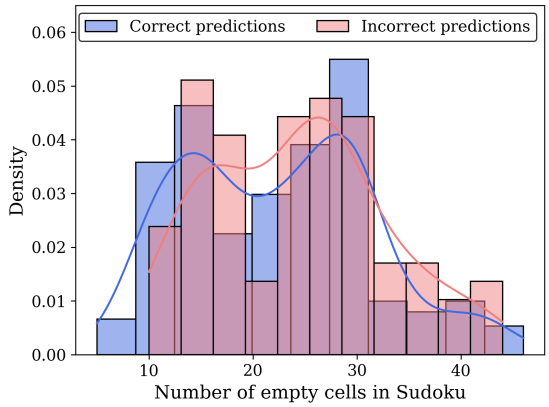


Figure 5: Density plot of number of empty cells for correct vs. incorrect predictions.

Model	Data	GSM8K	MATH
DeepSeek-R1-Distill-Qwen-1.5B	None	65.5	45.6
	Math-only	73.6	51.1
	Mixed	**76.1**	**53.1**
DeepSeek-R1-Distill-Qwen-7B	None	79.7	63.2
	Math-only	82.3	70.7
	Mixed	**87.4**	**71.4**

Table 5: Training with our puzzle data improves math reasoning on GSM8K and MATH.

doku, Graph Coloring, and Game of 24. We then train models on a mixture of this puzzle data and standard math data to test whether reasoning skills transfer beyond the puzzles themselves, ultimately improving overall mathematical reasoning.

Experimental Setup. Our puzzle answer is easy to verify, which is suitable for Reinforcement Learning. Specifically, we perform GRPO (Shao et al., 2024) on DeepSeek-R1-Distill-Qwen-1.5B and DeepSeek-R1-Distill-Qwen-7B (DeepSeek-AI et al., 2025). We prepare $10,000$ samples from our puzzle dataset, and another $15,000$ samples from MetaMathQA (Yu et al., 2024), a popular training set for mathematical reasoning. Other training details are reported in Appendix A.3.

Improvements on math benchmarks. We start with $2,000$ training samples, with a $1:1$ distribution between math and puzzle data. We compare with two baselines: first is the base model, second is training with entirely math samples. The results in Table 5 show that combining puzzle data with MetaMath yields the highest accuracy on both GSM8K and MATH for both models, outperforming training on MetaMath alone. This consistent performance improvement suggests that the state-checking and state-transition logic of puzzle-solving generalize to more general mathematical problems, aligning well with our initial hypothesis.

Optimal Ratio. To investigate the optimal ratio between puzzle-based and math-based data, we vary the proportion of math samples from 0.4 to 1.0 in a combined training set of 10k samples. In Figure 7, the performance on GSM8K steadily improves as the math ratio increases, peaking at a ratio of 0.8. Beyond this point, increasing the math ratio further results in lower accuracy. This suggests that neither pure math training nor pure puzzle training is optimal. Instead, a balanced combination of puzzle-based and traditional math data provides the best generalization. This indicates that our puzzle-based data though simple can complement the reasoning in standard math problems.

Scaling. We investigate the scaling effect of using math-only and mixed math-puzzle data. For the mixed data, we keep the ratio of math to 0.8. As we increase the number of training samples, both math-only and mixed approaches benefit from scaling up. Noticeably, the mixed approach consistently outperforms math-only. While math-only training shows diminishing returns or even slight

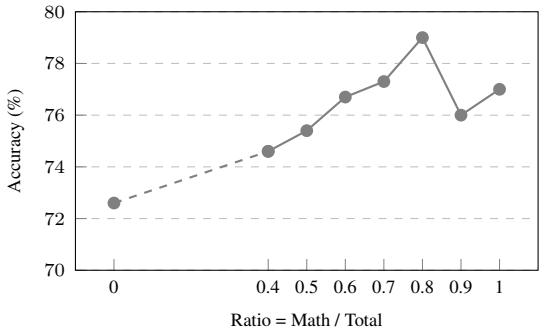


Figure 6: The effect of training data mixing (Math v.s. Puzzle) on the GSM8K.

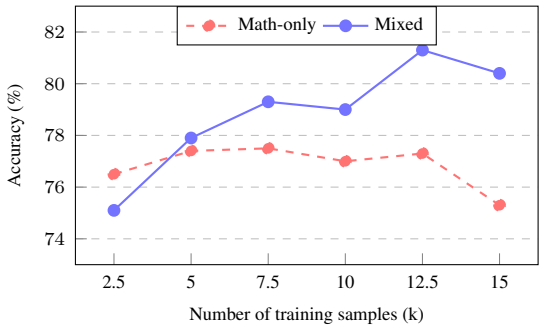


Figure 7: The scaling performance on GSM8K.

declines beyond 7.5k samples, the mixed approach continues to improve, reaching an accuracy peak of 81.3% with around 12.5k samples. This scaling phenomenon suggests the great potential of our simple puzzle data for enhancing the overall reasoning capability of LLMs.

6 Related Work

LLM Reasoning. Developing models to reason is a primary goal in natural language processing (Wos et al., 1992; Yang et al., 2018). Recently, LLMs, combined with prompting methods like CoT (Wei et al., 2022), Tree of Thought (Yao et al., 2023), and Self-Consistency (Wang et al., 2023) have shown remarkable performance across various reasoning tasks (Cobbe et al., 2021; Srivastava et al., 2022). For evaluation, current work primarily focuses on the final accuracy in reasoning-intensive domains like mathematics (Cobbe et al., 2021; Hendrycks et al., 2021; Chen et al., 2023; Rein et al., 2023; Ma et al., 2024), coding (Chen et al., 2021; Austin et al., 2021), commonsense (Mihaylov et al., 2018; Hendrycks et al., 2020), and logical reasoning (Yao et al., 2023; Long, 2023). However, as inference-time scaling gains importance (Snell et al., 2024; Guo et al., 2025), it's crucial to assess how effectively LLMs perform reflection and correction during reasoning. Tyagi et al. (2024) manually analyze LLMs' reasoning chains in logic puzzles, limiting scalability. Some studies (Singh et al., 2024; Zeng et al., 2024) evaluate LLMs' ability to handle reasoning mistakes, but the rule-based errors may be easily addressed by current LLMs. Moreover, they only statically assess LLMs reflection on past steps. In our work, we evaluate logic puzzles that significantly challenge current models and introduce two tasks that more closely reflect reasoning skills.

Puzzle Solving Tasks. Logic puzzles aim to deduce solutions from a set of rules (Giadikiaroglou et al., 2024). These puzzles hardly rely on prior knowledge of LLMs, making it ideal for thoroughly evaluating the reasoning capabilities (Li et al., 2024). Recent studies have explored LLMs on various puzzles with different emphases (Mittal et al., 2024). For example, Ishay et al. (2023); Long (2023) evaluate LLMs on Sudoku, challenging their numerical combination skills. Similarly, Ding et al. (2023); Yao et al. (2023) use the game of 24 to assess arithmetic calculations and strategic thinking. Other puzzles, such as grid puzzles (Dziri et al., 2024; Tyagi et al., 2024), crosswords (Yao et al., 2023), chess puzzles (Feng et al., 2024), mazes (Noever and Burdick, 2021), and Minesweeper (Li et al., 2024), have also been investigated. Nevertheless, the current evaluation of puzzles still focuses on final accuracy.

7 Conclusion

In this work, we introduced FINEREASON, a novel logic-puzzle benchmark designed to comprehensively evaluate the reasoning capabilities of LLMs. Unlike existing benchmarks that focus mainly on final-answer accuracy, FINEREASON delves into intermediate reasoning steps, specifically emphasizing state checking and transition actions. This fine-grained evaluation captures a model's ability to reflect, lookahead, and correct, which are vital aspects of human-like System 2 reasoning. Our experiments reveal significant gaps between reasoning-oriented and general-purpose LLMs, emphasizing the necessity to consider reflection and correction for robust reasoning evaluation. Furthermore, using puzzle-based data for training can enhance performance in broader mathematical tasks, highlighting the scalability of this approach and its potential to complement reasoning in standard math problems.

Limitations

Firstly, in our study, we employ the textual table as a representation of the puzzle state. Our evaluation shows that LLMs can reasonably understand such table format. Still, it is possible to explore other representation formats such as a coordinate or an image. Particularly, the image format can be further used to facilitate the evaluation of multimodal reasoning. This could be a possible future extension of our work.

Secondly, we employ zero-shot CoT (Kojima et al., 2022) as the prompt for generation in all our evaluations. While advanced prompting techniques such as Tree of Thoughts (Yao et al., 2023) have the potential to enhance performance, using them might shift the focus away from evaluating the genuine reasoning capabilities of LLMs without relying heavily on external assistance.

References

Arpit Agarwal, Katharina Muelling, and Katerina Fragkiadaki. 2019. Model learning for look-ahead exploration in continuous control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3151–3158.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. 2023. TheoremQA: A theorem-driven question answering dataset. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7889–7901, Singapore. Association for Computational Linguistics.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Antonia Creswell, Murray Shanahan, and Irina Higgins. 2023. Selection-inference: Exploiting large language models for interpretable logical reasoning. In *The Eleventh International Conference on Learning Representations*.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Ruomeng Ding, Chaoyun Zhang, Lu Wang, Yong Xu, Minghua Ma, Wei Zhang, Si Qin, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. 2023. Everything of thoughts: Defying the law of penrose triangle for thought generation. *arXiv preprint arXiv:2311.04254*.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, et al. 2024. Faith and fate: Limits of transformers on compositionality. *Advances in Neural Information Processing Systems*, 36.

Daniel Enström, Viktor Kjellberg, and Moa Johansson. 2024. Reasoning in transformers - mitigating spurious correlations and reasoning shortcuts. *CoRR*, abs/2403.11314.

Xidong Feng, Yicheng Luo, Ziyan Wang, Hongrui Tang, Mengyue Yang, Kun Shao, David Mguni, Yali Du, and Jun Wang. 2024. Chessgpt: Bridging policy learning and language modeling. *Advances in Neural Information Processing Systems*, 36.

Panagiotis Giadikiaroglou, Maria Lymperaiou, Giorgos Filandrianos, and Giorgos Stamou. 2024. Puzzle solving using reasoning of large language models: A survey. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11574–11591, Miami, Florida, USA. Association for Computational Linguistics.

Google. 2024. Gemini-2.0-flash family.

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

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Adam Ishay, Zhun Yang, and Joohyung Lee. 2023. Leveraging large language models to generate answer set programs. *arXiv preprint arXiv:2307.07699*.

Daniel Kahneman. 2011. *Thinking, fast and slow*. Farrar, Straus and Giroux.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*.

Yinghao Li, Haorui Wang, and Chao Zhang. 2024. Assessing logical puzzle solving in large language models: Insights from a minesweeper case study. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 59–81, Mexico City, Mexico. Association for Computational Linguistics.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Jieyi Long. 2023. Large language model guided tree-of-thought. *arXiv preprint arXiv:2305.08291*.

Jingkun Ma, Runzhe Zhan, Derek F. Wong, Yang Li, Di Sun, Hou Pong Chan, and Lidia S. Chao. 2024. Visaidmath: Benchmarking visual-aided mathematical reasoning. *CoRR*, abs/2410.22995.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

Chinmay Mittal, Krishna Kartik, Parag Singla, et al. 2024. Puzzlebench: Can llms solve challenging first-order combinatorial reasoning problems? *arXiv preprint arXiv:2402.02611*.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*.

David A. Noever and Ryerson Burdick. 2021. Puzzle solving without search or human knowledge: An unnatural language approach. *ArXiv*, abs/2109.02797.

OpenAI. 2022. Gpt3.5 turbo.

OpenAI. 2024. Learning to reason with llms.

OpenAI, Aaron Hurst, Adam Lerer, Adam P. Goucher, et al. 2024. GPT-4o system card. *ArXiv*, abs/2410.21276.

Yiwei Qin, Xuefeng Li, Haoyang Zou, Yixiu Liu, Shijie Xia, Zhen Huang, Yixin Ye, Weizhe Yuan, Hector Liu, Yuanzhi Li, et al. 2024. O1 replication journey: A strategic progress report–part 1. *arXiv preprint arXiv:2410.18982*.

An Yang Qwen, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 technical report. *ArXiv*, abs/2412.15115.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*.

Rebecca Roelofs, Vaishaal Shankar, Benjamin Recht, Sara Fridovich-Keil, Moritz Hardt, John Miller, and Ludwig Schmidt. 2019. A meta-analysis of overfitting in machine learning. *Advances in Neural Information Processing Systems*, 32.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Joykirat Singh, Akshay Nambi, and Vibhav Vineet. 2024. Exposing the achilles' heel: Evaluating llms ability to handle mistakes in mathematical reasoning. *arXiv preprint arXiv:2406.10834*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

Kimi Team. 2025. Kimi k1.5: Scaling reinforcement learning with llms. *ArXiv*, abs/2501.12599.

Nemika Tyagi, Mihir Parmar, Mohith Kulkarni, Aswin Rrv, Nisarg Patel, Mutsumi Nakamura, Arindam Mitra, and Chitta Baral. 2024. Step-by-step reasoning to solve grid puzzles: Where do LLMs falter? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19898–19915. Association for Computational Linguistics.

Peter van Beek. 2006. Chapter 4 - backtracking search algorithms. In Francesca Rossi, Peter van Beek, and Toby Walsh, editors, *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*, pages 85–134. Elsevier.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Zihan Wang, Xiangyang Li, Jiahao Yang, Yeqi Liu, Junjie Hu, Ming Jiang, and Shuqiang Jiang. 2024. Lookahead exploration with neural radiance representation for continuous vision-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13753–13762.

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

Larry Wos, Ross Overbeek, Ewing Lusk, and Jim Boyle. 1992. *Automated reasoning introduction and applications*. McGraw-Hill, Inc.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng YU, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024. Metamath: Bootstrap your own mathematical questions for large language models. In *The Twelfth International Conference on Learning Representations*.

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

Zhongshen Zeng, Yinhong Liu, Yingjia Wan, Jingyao Li, Pengguang Chen, Jianbo Dai, Yuxuan Yao, Rongwu Xu, Zehan Qi, Wanru Zhao, Linling Shen, Jianqiao Lu, Haochen Tan, Yukang Chen, Hao Zhang, Zhan Shi, Bailin Wang, Zhijiang Guo, and Jiaya Jia. 2024. MR-ben: A meta-reasoning benchmark for evaluating system-2 thinking in LLMs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

A Appendix

A.1 Dataset Statistics

Figure 8 presents the statistics for the four tasks, including the total number of questions, as well as the number of solvable and unsolvable states for each task. For grid puzzles, we can only sample 94 solvable states with unsolvable children, resulting in a somewhat imbalanced dataset. Nonetheless, we have maintained balance between solvable and unsolvable states for the remaining three puzzles.

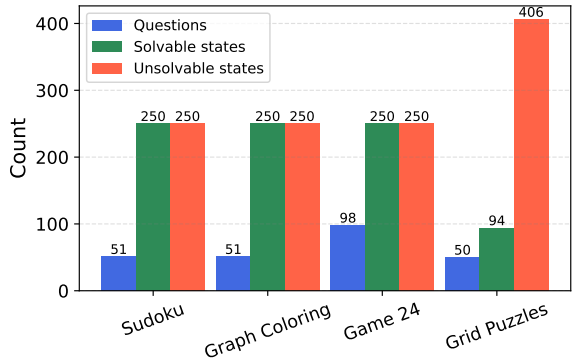


Figure 8: Dataset statistics

A.2 Prompt Templates

Table 8 shows the state checking and state transition prompts for a Sudoku example.

A.3 Training Details

We train our models using GRPO based on OpenR1[2]. We use one GPU to run vLLM for faster generation and the remaining GPUs for training. The hyperparameters and training details are reported in Table 6.

Learning rate	4e-5
Warm up ratio	0.1
Batch size	112
Max prompt length	1024
Max completion length	1024
Training epochs	1
Hardware	8 H20 (96GB)

Table 6: Hyperparameter and training details.

A.4 Additional Experimental Results

Table 7 reports the state-checking precision, recall, and F1 scores of models across four tasks. It is observed that o1 consistently outperforms all other models in detecting unsolvable states, as evidenced by the high recall and precision acorss tasks. Models like GPT-4o, Qwen2.5-Inst and Gemini-F are generally more precise when they predict unsolvability, but are limited by low recall in tasks like Sudoku and Grid Puzzles. GPT-3.5 generally struggles with both recall and precision, especially in more complex tasks like Sudoku.

Puzzle	Model	Recall	Precision	F1
Sudoku	o1	73.2	86.7	79.4
	GPT-4o	6.4	80.0	11.9
	GPT-3.5	28.0	49.0	35.6
	Gemini-FT	87.2	64.3	74.0
	Gemini-F	3.2	57.1	6.06
	Qwen2.5-Inst	4.8	75.0	9.02
Game of 24	o1	95.6	99.2	97.4
	GPT-4o	95.6	75.9	84.6
	GPT-3.5	54.8	56.6	55.7
	Gemini-FT	94.8	97.5	96.1
	Gemini-F	98.8	89.2	93.7
	Qwen2.5-Inst	97.6	82.2	89.2
Graph Coloring	o1	93.1	95.9	94.5
	GPT-4o	44.8	57.8	50.5
	GPT-3.5	27.4	53.5	36.3
	Gemini-FT	96.8	89.2	92.8
	Gemini-F	29.0	64.3	40.0
	Qwen2.5-Inst	25.8	73.6	38.2
Grid Puzzles	o1	93.8	92.5	93.2
	GPT-4o	47.8	88.2	62.0
	GPT-3.5	39.4	79.6	52.7
	Gemini-FT	91.6	94.7	93.1
	Gemini-F	24.4	94.3	38.7

Table 7: Precision, Recall and F1 scores of state checking task for all puzzles.

[2]https://github.com/huggingface/open-r1

State Checking

You are given a partially filled 9x9 Sudoku grid represented as a list of lists, where empty cells are represented as 0. Your task is to determine if this current state can lead to a solvable solution. Specifically, use lookahead techniques to determine if it's possible to fill the remaining cells according to standard Sudoku rules, ensuring that each row, column, and 3x3 subgrid contains unique numbers from 1 to 9.
Additionally, you are provided with a previously explored next state that has been proven to be unsolvable. Use this information to avoid revisiting this failed path and leverage it to make a more informed decision about the current state.

Current state: [[4, 1, 6, 9, 7, 2, 8, 3, 5], [7, 2, 3, 1, 8, 5, 6, 9, 4], [5, 9, 8, 3, 4, 6, 2, 1, 7], [6, 3, 5, 4, 1, 9, 7, 2, 8], [1, 8, 9, 2, 6, 7, 4, 5, 3], [2, 4, 7, 5, 3, 8, 1, 6, 9], [8, 7, 2, 6, 9, 3, 5, 4, 1], [3, 6, 0, 0, 5, 0, 0, 7, 2], [0, 0, 0, 0, 0, 4, 3, 8, 0]]
Explored next state that leads to an unsolvable path: [[4, 1, 6, 9, 7, 2, 8, 3, 5], [7, 2, 3, 1, 8, 5, 6, 9, 4], [5, 9, 8, 3, 4, 6, 2, 1, 7], [6, 3, 5, 4, 1, 9, 7, 2, 8], [1, 8, 9, 2, 6, 7, 4, 5, 3], [2, 4, 7, 5, 3, 8, 1, 6, 9], [8, 7, 2, 6, 9, 3, 5, 4, 1], [3, 6, 1, 0, 5, 0, 0, 7, 2], [0, 0, 0, 0, 0, 4, 3, 8, 0]]
Let's think step by step, considering the failed state to avoid unnecessary exploration. Do not solve using programming.
Choose from (A) Solvable (B) Unsolvable. End your answer with "Answer: (A)" or "Answer: (B)".

State Transition

You are given an initial Sudoku puzzle S(0), followed by a sequence of progressive states leading to the current state S(i). Alongside each state, its solvability status L(*) is given. Your task is to determine the next state by making exactly one move, ensuring progress toward a valid solution. A valid Sudoku solution requires that each row, column, and 3x3 subgrid contains the numbers 1 to 9 without repetition.
Additionally, you are provided with a previously explored next state that has been proven to be unsolvable. Use this information to avoid revisiting this failed path.
A move is defined as either:
1. Filling: Replacing a 0 in exactly one empty cell with a value from 1 to 9.
2. Removing: Replacing a value in exactly one filled cell with 0.

Initial puzzle:
S(0) = [[0, 1, 0, 0, 7, 0, 8, 3, 0], [0, 2, 0, 0, 0, 0, 6, 0, 4], [5, 9, 0, 0, 0, 0, 0, 1, 0], [0, 0, 5, 0, 1, 9, 0, 0, 0], [1, 0, 0, 2, 6, 0, 0, 5, 3], [2, 4, 7, 0, 0, 8, 0, 0, 9], [0, 7, 0, 6, 9, 0, 0, 0, 1], [3, 0, 0, 0, 5, 0, 0, 7, 2], [0, 0, 0, 0, 0, 4, 3, 8, 0]]
L(0) = Solvable
Two moves ago:
S(i-2) = [[4, 1, 6, 9, 7, 2, 8, 3, 5], [7, 2, 3, 1, 8, 5, 6, 9, 4], [5, 9, 8, 3, 4, 6, 2, 1, 7], [6, 3, 5, 4, 1, 9, 7, 2, 8], [1, 8, 9, 2, 6, 7, 4, 5, 3], [2, 4, 7, 5, 3, 8, 1, 6, 9], [8, 7, 2, 6, 9, 3, 5, 0, 1], [3, 0, 0, 0, 5, 0, 0, 7, 2], [0, 0, 0, 0, 0, 4, 3, 8, 0]]
L(i-2) = Solvable
One move ago:
S(i-1) = [[4, 1, 6, 9, 7, 2, 8, 3, 5], [7, 2, 3, 1, 8, 5, 6, 9, 4], [5, 9, 8, 3, 4, 6, 2, 1, 7], [6, 3, 5, 4, 1, 9, 7, 2, 8], [1, 8, 9, 2, 6, 7, 4, 5, 3], [2, 4, 7, 5, 3, 8, 1, 6, 9], [8, 7, 2, 6, 9, 3, 5, 4, 1], [3, 0, 0, 0, 5, 0, 0, 7, 2], [0, 0, 0, 0, 0, 4, 3, 8, 0]]
L(i-1) = Solvable
Current state:
S(i) = [[4, 1, 6, 9, 7, 2, 8, 3, 5], [7, 2, 3, 1, 8, 5, 6, 9, 4], [5, 9, 8, 3, 4, 6, 2, 1, 7], [6, 3, 5, 4, 1, 9, 7, 2, 8], [1, 8, 9, 2, 6, 7, 4, 5, 3], [2, 4, 7, 5, 3, 8, 1, 6, 9], [8, 7, 2, 6, 9, 3, 5, 4, 1], [3, 6, 0, 0, 5, 0, 0, 7, 2], [0, 0, 0, 0, 0, 4, 3, 8, 0]]
L(i) = Solvable
Explored next state:
S(i+1) = [[4, 1, 6, 9, 7, 2, 8, 3, 5], [7, 2, 3, 1, 8, 5, 6, 9, 4], [5, 9, 8, 3, 4, 6, 2, 1, 7], [6, 3, 5, 4, 1, 9, 7, 2, 8], [1, 8, 9, 2, 6, 7, 4, 5, 3], [2, 4, 7, 5, 3, 8, 1, 6, 9], [8, 7, 2, 6, 9, 3, 5, 4, 1], [3, 6, 1, 0, 5, 0, 0, 7, 2], [0, 0, 0, 0, 0, 4, 3, 8, 0]]
L(i+1) = Unsolvable
Let's think step by step. Analyze the progress made so far and determine the immediate next move. End your answer with "Next state: {grid}", where {grid} is in the same python list format as the previous states.

Table 8: Prompt templates for state checking and state transition in Sudoku.