

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

Guizhen Chen^{1,2,*} Weiwen Xu^{2,†} Hao Zhang^{2,†} Hou Pong Chan²
Chaoqun Liu^{1,2,*} Lidong Bing² Deli Zhao^{2,3} Anh Tuan Luu¹ Yu Rong^{2,3}

¹ Nanyang Technological University, Singapore

² DAMO Academy, Alibaba Group, Singapore

³ Hupan Lab, Hangzhou, China

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 systematic 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 general reasoning. We show that models trained on our state checking and transition data demonstrate gains in mathematical reasoning by up to 5.1%.¹

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 suggest that large language models (LLMs) can

*Guizhen and Chaoqun are under the Joint PhD Program between Alibaba and NTU.

†Corresponding authors.

¹<https://github.com/DAMO-NLP-SG/FineReason>

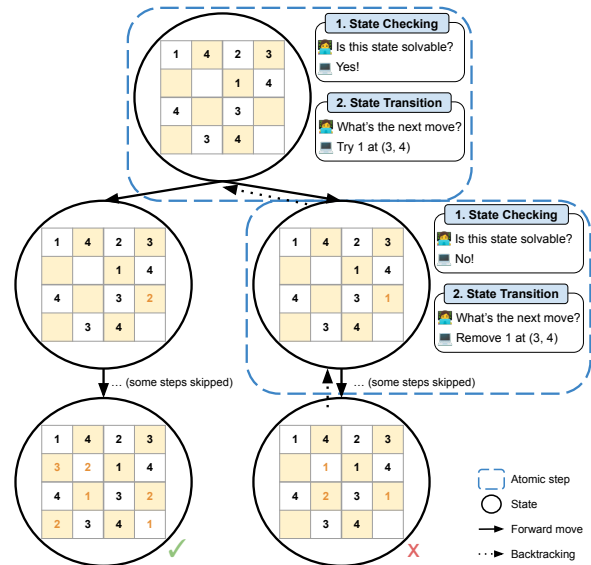


Figure 1: A simplified Sudoku solution tree illustrating stepwise state checking (solvability prediction) and transition (next move determination, including forward moves and backtracking).

attain advantages akin to those of System 2 reasoning (OpenAI, 2024; Snell et al., 2025; DeepSeek-AI et al., 2025; Team, 2025). Instead of generating answers directly, these models can iteratively reflect and correct their reasoning (Shinn et al., 2023), achieving strong performance on reasoning-intensive tasks like mathematics and coding (Qin et al., 2024; Li et al., 2024a; Muennighoff et al., 2025).

Despite these improvements, the underlying mechanisms remain underexplored. Existing reasoning benchmarks primarily focus on the final-answer accuracy (Hendrycks et al., 2021a; Cobbe et al., 2021; Chen et al., 2021; Hendrycks et al., 2021b), which offers limited insight into LLMs’ internal reasoning processes, such as reflection and correction. For instance, models might reach a correct conclusion through flawed reasoning (Zelikman et al., 2022; Creswell et al., 2023; Lightman

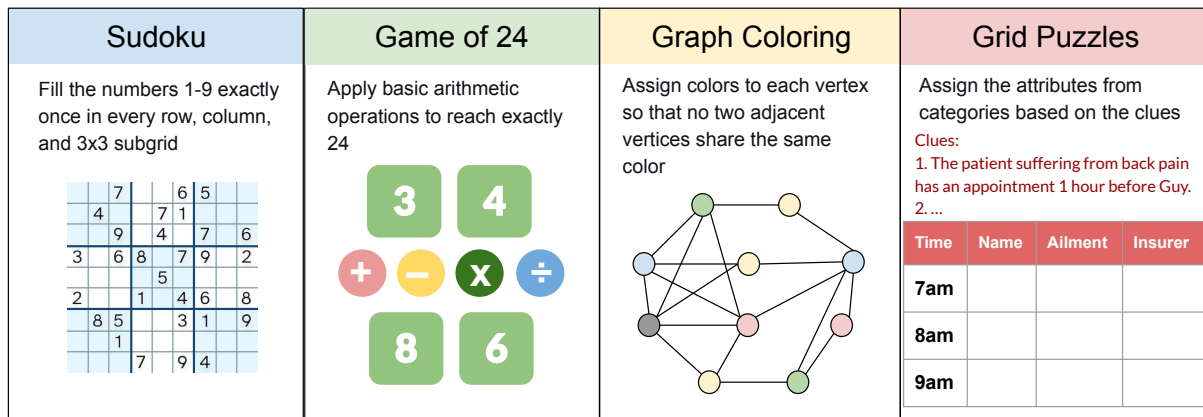


Figure 2: An illustration of four puzzle categories in FINEREAISON. These puzzles are solved through discrete steps, with explicit rules allowing easy validation of intermediate states.

et al., 2024; Chia et al., 2024a). This diminishes the trustworthiness of model outputs, posing potential risks in practical usage. Moreover, models might achieve high accuracy by exploiting superficial patterns in the training data (Roelofs et al., 2019; Xu et al., 2023; Enström et al., 2024), making it difficult to ascertain whether the observed performance gain truly stems from deliberate reasoning. Therefore, there is a pressing need for more comprehensive reasoning benchmarks that assess the integrity of intermediate processes.

In this work, we present FINEREAISON, a logic-puzzle benchmark for evaluating LLMs’ deliberate reasoning capabilities, such as reflection, backtracking, and exploring alternative solutions. FINEREAISON includes four types of puzzles: *Sudoku*, *Graph Coloring*, *Game of 24*, and *Grid Puzzles*, as shown in Figure 2. Solving a logic puzzle involves a series of discrete steps, and the explicit rules make it straightforward to validate the intermediate states. To structure our evaluation, we propose two key actions in each atomic step, **state checking** and **state transition**, as illustrated 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 (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 reveals a significant 19.7% per-

formance gap between OpenAI-o1 (OpenAI, 2024) and Gemini-2.0-Flash-Thinking (Google, 2024), a difference not captured by other maths and coding benchmarks where high scores saturate. General-purpose models like GPT-4o (OpenAI et al., 2024) struggle with deliberate reasoning in FINEREAISON, making near-random guesses in state checking and poor performance in state transition.

To enhance reasoning, we develop a specialized training set on puzzle state checking and transition. Integrating our dataset with common reasoning data consistently improves performance on complex reasoning tasks. For example, when applied to DeepSeek-R1-Distill-Qwen-7B, our puzzle data boosts the accuracy from 82.3% to 87.4% on GSM8K, compared to models trained exclusively on math data. Our results suggest that skills such as backtracking and constraint validation generalize from puzzles to general reasoning, similar to how structured practice (e.g., chess) enhances human strategic thinking and problem solving.

Our main contributions are three-fold: 1) We introduce FINEREAISON, a puzzle-based benchmark accompanied by systematic evaluations on state checking and transition, to provide a more precise evaluation of models’ reasoning abilities, particularly in reflection and correction. 2) Experimental results reveal substantial limitations in deliberate reasoning among general-purpose models, and even in the leading reasoning models. 3) We show that training on structured puzzles transfers deliberate reasoning skills to general problem-solving.

2 FINEREAISON

We present FINEREAISON, a logic-puzzle benchmark that comprehensively assesses LLMs’ reason-

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 FINERREASON.

ing through stepwise evaluation of state checking and transition. Inspired by the adage “think twice before acting,” these actions capture how models assess the current situation (*i.e.*, state checking) and plan the next move (*i.e.*, state transition), skills crucial for deliberate reasoning.

Formally, the reasoning process of a logic puzzle can be represented as $p = \{p_1, p_2, \dots, p_n\}$, 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 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 in our puzzles (§2.1). Next, we describe the two key actions – state checking and state transition – that facilitate reasoning evaluation of models (§2.2).

2.1 Tree-based Puzzle Decomposition

Puzzle solving can be represented as a tree, where nodes are intermediate states, 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 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 different 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. In Sudoku, the aim is to fill the empty cells such that each row, each column, and each

of the 3×3 subgrids contains all digits from 1 to 9 exactly once. A Sudoku state can be either a partially or fully filled 9×9 grid. The minimal move is defined as either adding a number for exploration or removing a number for backtracking. We use Sudoku questions from Kaggle² to create the Sudoku trees.

Graph Coloring. The aim 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 or a completely colored graph. 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) 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 problem set from Yao et al. (2023) to generate the trees.

Logic Grid Puzzles. Logic grid puzzles involve filling a grid with attributes from multiple categories (*e.g.*, color, time) based on a set of textual clues. Each attribute must appear exactly once per category and satisfy all given clues. Each state is a partially or fully filled grid, with minimal moves being adding or removing attribute combinations specified in a clue. Our grid puzzle trees are constructed from Tyagi et al. (2024). Unlike other puzzles, translating textual clues into verification code is challenging, especially when it involves attribute comparisons. To address this, we define three functions: $r(v)$ and $c(v)$ to retrieve the row and column numbers of an attribute v , and $T(i, j)$

²<https://www.kaggle.com/datasets/bryanpark/sudoku>

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 use one-shot prompting with GPT-4o (OpenAI et al., 2024) for clue translation, followed by manual verification. We ensure all solutions pass the coded clues.

2.2 Evaluation Tasks

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

State Checking. State checking requires the model to assess if a given state s_i can lead to a solvable solution s_n . Based on our constructed puzzle trees, 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 models can reflect on prior steps 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 unsolvable states.

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 ground-truth state-checking labels. We sample states from the puzzle tree and construct prompts with puzzle rules, prior visited states, the current 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 on 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.

Implementation. We use zero-shot chain of thought prompting (Kojima et al., 2022) for evaluation. To ensure a genuine evaluation of LLM’s inherent reasoning capabilities, we explicitly include “Do not solve using programming” in the instruction, restricting models from relying on memorized code snippets from their training data. In our actual attempts, without explicitly stating this constraint, models tend to generate Sudoku solvers or graph-coloring algorithms instead of demonstrating deliberate reasoning. Prompt templates are shown in Appendix A.2.

Models. We select the best-performing open and closed-source models, including 1) reasoning-oriented models: o1 (OpenAI, 2024), Gemini-2.0-Flash-Thinking (Gemini-FT, Google (2024)), and 2) general-purpose 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 Qwen2.5-72B-Instruct (Qwen2.5-Inst, Qwen et al. (2025)).

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.

4 Evaluation Results

In this section, we first present a preliminary evaluation of LLMs on end-to-end puzzle solving (§4.1), which reveals inconsistencies in model performance. To gain deeper insights beyond their end-to-end performance, we present our main results focusing on state checking and transition tasks (§4.2), followed by analyses on models’ behaviors, error patterns, and performance across different difficulty levels (§4.3 to §4.6).

4.1 Preliminary: End-to-End Puzzle Solving

Table 2 presents an initial evaluation of LLMs on end-to-end puzzle-solving tasks. Despite their strong performance on coding and math tasks (Qwen et al., 2025), these models are notably weak in end-to-end puzzle solving. Additionally, there are some counter-intuitive observations: Gemini-F outperforms Gemini-FT on Sudoku and Game of 24, yet struggles on the other two puzzles. These inconsistencies suggest that end-to-end puzzle solving alone may not be a reliable metric for assessing LLMs’ reasoning, emphasizing the need for more granular evaluation methods.

4.2 Main results

To understand models’ reasoning capabilities in greater depth, we evaluate on state checking and transition. Results in Table 3 reveal noticeable performance gaps between reasoning-oriented and general-purpose models. On the state-checking task, reasoning-oriented models (o1 and Gemini-FT) consistently lead the performance in every puzzle category. In contrast, general-purpose models barely match the random baseline in puzzles like Sudoku and Grid Puzzles. A similar trend is observed on the state-transition task. These findings further support the view that inference-time scaling can substantially boost reasoning capabilities (Snell et al., 2025; Muennighoff et al., 2025).

Between the reasoning models, Gemini-FT generally performs on par with o1 in state checking but consistently lags behind in state transition. This reveals weaknesses in Gemini-FT’s reasoning process, particularly in error revision. These findings align well with our practical experience using these models, which provides empirical evidence that FINEREASON offers a more accurate reflection of LLMs’ reasoning capabilities.

4.3 Model Behaviors in State Checking

State checking requires looking ahead to identify unsolvable states. To analyze models’ behaviors, we report the state-checking precision, recall, and F1 scores in Table 4, designating unsolvable states as positive cases. Recall measures the model’s ability to detect dead ends, while precision indicates the reliability of its unsolvable state predictions.

We find that reasoning models generally detect unsolvable states well, as indicated by the high F1 scores. As for general models (GPT-4o, Gemini-F, Qwen2.5-Inst), the recall is generally low in Sudoku but not in Game of 24, possibly due to Sudoku’s deeper puzzle tree making unsolvable state detection harder. A similar trend is also observed in other puzzles that have deeper solutions, including graph coloring and grid puzzles (see Appendix A.3). These observations reveal that general LLMs tend to make overly optimistic decisions (i.e., assuming a solvable state) when faced with tasks that exceed their actual capabilities.

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

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.

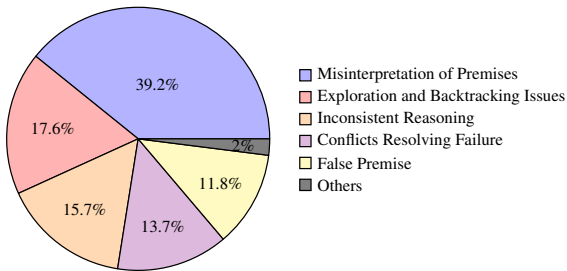


Figure 3: Human analysis of error types.

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 the 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,” the model still mistakenly assigned Joey’s cake to the \$125 cake. Additionally, the model might fail to explore alternative paths, leading to an incorrect assessment of current states. 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. Examples of the errors are shown in Appendix A.4.

4.5 Model Behaviors in State Transition

To understand models’ behaviors during state transition, we break down the performance by class and count the common mistakes made by models.

The left chart of Figure 4 shows the Sudoku state-transition performance breakdown for each class (solvable vs. unsolvable). We 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 might be attributed to a forward-generation reasoning style of LLMs. This trend, however, does not apply to GPT-3.5, which shows significantly weak performance and tends to rely primarily on random guessing.

The right chart of Figure 4 shows the errors typically made by models during state transition. At solvable states, common errors include making multiple moves (Multiple Moves), violating Sudoku rules (Invalid Move), and transitioning to an unsolvable child state (Unsolvable Child). At 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). Examples of the errors are shown in Appendix A.5. 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 state, indicating that LLMs struggle with step-by-step backtracking. For reasoning models (o1 and Gemini-FT), transitioning to siblings is the second most frequent error. This error is due to violating the minimal move principle (Table 1), highlighting weaknesses in their instruction-following capability. For general models, they 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 of correct vs. incorrect predictions by the number of unfilled cells in Sudoku states (Figure 5). We find that 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, and the proportion of incorrect predictions increases. This likely reflects the increased computational complexity of looking ahead and determining solvability when many possibilities exist.

5 Training on puzzle data

As highlighted in §4, most models struggle with identifying dead ends and backtracking. These observations reveal critical bottlenecks in models’ reflection and correction abilities, which are essen-

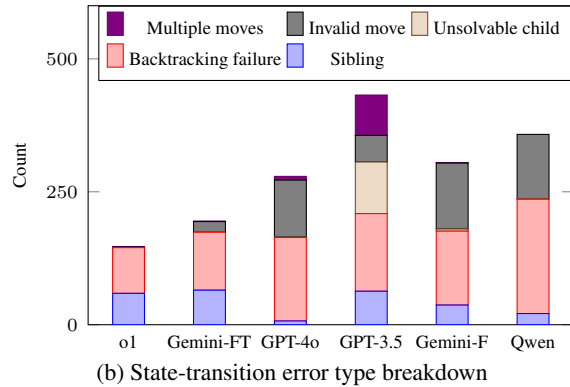
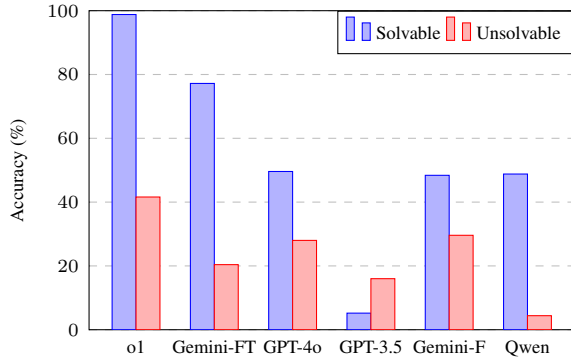


Figure 4: Performance breakdown and error analysis of state-transition in Sudoku.

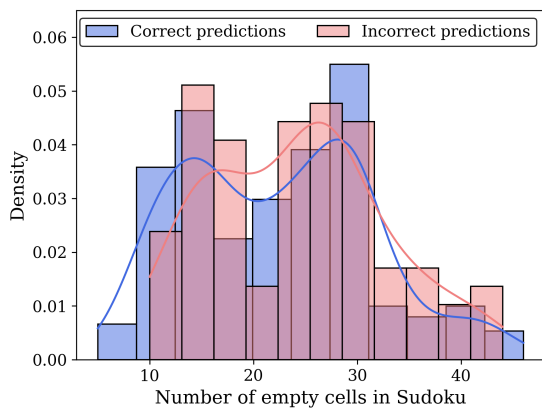


Figure 5: Density plot of number of empty cells for correct vs. incorrect predictions.

tial for reasoning. We hypothesize that training on state-checking and state-transition tasks can provide synthetic reflection, allow models to practice error correction, and ultimately transfer these reasoning skills to mathematical problem solving.

To validate our hypothesis, we construct a training set consisting of state-checking and state-transition data from Sudoku, Graph Coloring, and Game of 24. We train models on a mixture of puzzle data and math data to test whether reasoning skills transfer beyond puzzles themselves, ultimately improving performance on math tasks.

5.1 Experimental setup

We prepare 10,000 puzzle samples and another 15,000 samples from MetaMathQA (Yu et al., 2024), a popular training set for mathematical reasoning. The puzzle states are easily verified, making it suitable for Reinforcement Learning. Specifically, we train GRPO (Shao et al., 2024) on DeepSeek-R1-Distill-Qwen-1.5B and DeepSeek-R1-Distill-Qwen-7B (DeepSeek-AI et al., 2025).

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.

Other training details are reported in Appendix A.6. Due to limited computing resources, we restrict the maximum completion length to 1024 in both training and evaluation.

5.2 Improvements on mathematical reasoning

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

To further assess the impact of the mixed training, we analyze the number of correctly solved samples at each difficulty level in the MATH dataset, where a higher level means higher difficulty. Results in Appendix A.7 indicate that mixed training is especially effective for harder problems (levels 4 and 5), likely because complex problems require more reflection and correction. By incorporating puzzle-solving data into math reasoning training, we encourage models to reflect and backtrack when

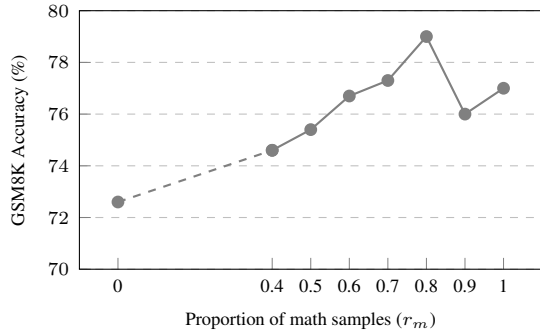


Figure 6: The effect of math ratio on the 1.5B model.

Data	GSM8K
None	65.5
Math-only	73.6
Mixed ($r_u = 0.2$)	74.8
Mixed ($r_u = 0.5$)	76.1
Mixed ($r_u = 0.8$)	77.4

Table 6: The effect of unsolvable ratio on the 1.5B model.

necessary.

To investigate the nature of improvements, we examine corrected examples after training. Around 60% of these examples use longer reasoning steps, determined by sentence splitter. Through quality analysis of 20 examples, we find more evidence of verification and self-correction in the thought process. Examples are shown in Appendix A.8.

5.3 Effect of the puzzle ratio

To study the optimal ratio between puzzle-based and math-based data, we vary the proportion of math samples, r_m , from 0.4 to 1.0 in a combined training set of $10k$ samples. In Figure 6, 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 math data provides the best performance. This indicates that our puzzle-based data, though simple, can complement the reasoning in standard math problems.

In addition, our previous analysis shows that models struggle significantly more with unsolvable states than solvable states in both state checking (§4.3) and state transition (§4.5). Our error analysis also shows that “backtracking failure” is a major error (§4.4), even for top reasoning mod-

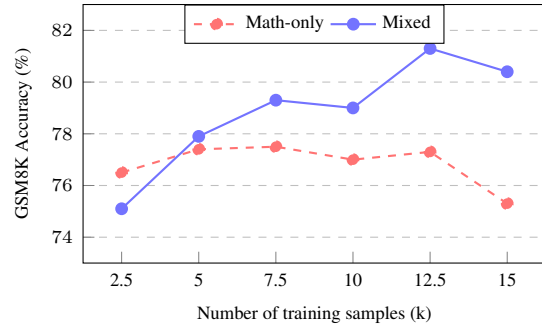


Figure 7: The scaling performance of the 1.5B model.

els like o1. This means we could prioritize the learning for backtracking from these unsolvable cases to maximize learning benefits. To validate this, we examine the effect of varying the ratio of unsolvable data, r_u , in mixed training. In Table 6, it is evident that increasing the proportion of unsolvable data boosts the performance on GSM8K, suggesting that focusing on these unsolvable cases enhances the model’s ability to detect dead ends and revise strategies—skills transferable to math problem-solving.

5.4 Analysis on the scaling effect

We examine the scaling effect of using math-only and mixed data for training. We use the optimal math ratio ($r_m = 0.8$) in the mixed data. The results in Figure 7 show that, as we increase the number of training samples, both approaches benefit from scaling up. Noticeably, the mixed approach consistently outperforms math-only training starting from $5k$ samples. While math-only training shows diminishing returns or even a slight decline beyond $7.5k$ samples, the mixed approach continues to improve, reaching an accuracy peak of 81.3% with around $12.5k$ samples. This scaling effect suggests the great potential of our simple puzzle data for enhancing the overall reasoning capability of LLMs.

5.5 Improvements on general reasoning

To investigate if our puzzle data benefits general reasoning tasks beyond math, we conduct additional experiments on MMLU (Hendrycks et al., 2021a), a dataset covering 57 tasks including STEM, humanities, social sciences, and others. Using the same optimal ratio ($r_m = 0.8$) as in math training, we observe consistent performance improvements over MMLU-only training (Table 7).

Next, we analyze the performance improvement across different subsets in MMLU. Results in Ap-

Model	Data	MMLU-test
DeepSeek-R1-Distill-Qwen-1.5B	None	37.8
	MMLU-train-only	41.1
	Mixed	44.9
DeepSeek-R1-Distill-Qwen-7B	None	54.9
	MMLU-train-only	61.2
	Mixed	64.2

Table 7: Training with our puzzle data improves general reasoning tasks on MMLU.

pendix A.9 show that mixed training significantly improves performance on complex, multi-step reasoning tasks that might require more reflection (STEM +10%, Social Sciences +3.8%), but shows less benefit for simpler, more direct question types.

Overall, our puzzle data bridges the gap of the lack of annotated intermediate training data for general reasoning tasks by providing synthetic, structured scenarios for learning reasoning skills. While real-world reasoning is less deterministic, our results indicate that mastering deterministic steps in puzzle solving builds foundational skills like reflection and correction, which demonstrably transfer to general reasoning tasks.

6 Related Work

Reasoning Capabilities of LLMs. Advancing the reasoning capabilities of large language models is a critical goal in natural language processing (Wos et al., 1992; Yang et al., 2018). In recent years, LLMs, combined with prompting techniques such as Chain of Thought (Wei et al., 2022), Tree of Thought (Yao et al., 2023), and Self-Consistency (Wang et al., 2023), have shown remarkable performance in various reasoning tasks (Cobbe et al., 2021; Srivastava et al., 2023). Current evaluation methods focus mainly on the final accuracy in reasoning-intensive domains, including mathematics (Cobbe et al., 2021; Hendrycks et al., 2021b; Chen et al., 2023; Rein et al., 2024; Ma et al., 2024), science (Mihaylov et al., 2018; Xu et al., 2021a,b; Huang et al., 2025, 2024), coding (Chen et al., 2021; Austin et al., 2021), commonsense (Hendrycks et al., 2021a), and logical reasoning (Yao et al., 2023; Long, 2023). However, as inference-time scaling gains importance (Snell et al., 2025; DeepSeek-AI et al., 2025) and models become more capable of reasoning, it is crucial to assess how effectively models perform reflection and correction during reasoning. While Tyagi et al. (2024) manually analyzes the reasoning chains in logic puzzles, their approach lacks

scalability. Some studies (Singh et al., 2024; Zeng et al., 2024; Xu et al., 2024) evaluate how models handle reasoning mistakes, but these investigations often rely on rule-based mistakes that may be easily resolved by current LLMs. Moreover, these studies only assess reflection on past steps in a static manner. In our work, we address these limitations by introducing two novel tasks designed to more accurately reflect models’ capabilities in dynamic reasoning and error correction.

Puzzle Solving Tasks. Logic puzzles, which require deducing solutions from a set of rules (Giadikiaroglou et al., 2024), are ideal for evaluating the reasoning abilities of LLMs, as they rely minimally on prior knowledge (Li et al., 2024b). Recent studies have explored LLMs on various puzzles with different emphases (Mittal et al., 2024), such as Sudoku (Ishay et al., 2023; Long, 2023) for strategic thinking, Game of 24 (Ding et al., 2024; Yao et al., 2023) for arithmetic calculations. Some investigate grid puzzles (Dziri et al., 2023; Tyagi et al., 2024), crosswords (Yao et al., 2023), chess puzzles (Feng et al., 2023), mazes (Noever and Burdick, 2021), Minesweeper (Li et al., 2024b), and abstract pattern recognition (Chia et al., 2024b). However, the evaluation remains mainly focused on the final accuracy.

7 Conclusion

In this work, we introduce 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 the model’s ability to reflect, look ahead, and backtrack, which are vital aspects of human-like System 2 reasoning. Our experiments reveal significant gaps between reasoning-oriented and general-purpose LLMs, emphasizing the need to consider reflection and correction for robust reasoning evaluation. Furthermore, using puzzle-based data for training can enhance performance in general reasoning tasks, highlighting the scalability of this approach and its potential to transfer reasoning skills beyond puzzles to broader reasoning.

Acknowledgment

This research is supported by DAMO Academy through DAMO Academy Research Intern Program and Alibaba-NTU Singapore Joint Research Institute (JRI), Nanyang Technological University, Singapore. We would also like to extend our gratitude to Interdisciplinary Graduate Programme and College of Computing and Data Science of NTU, for their support.

Limitations

Our study has several limitations. First, we represent puzzle states using textual tables. Our evaluation shows that models can reasonably understand this table format. However, there is potential to explore alternative representation formats, such as coordinates or images. The image format could be particularly valuable for facilitating the evaluation of multi-modal reasoning, offering a promising direction for future extensions of our work. Second, we employ zero-shot CoT prompting (Kojima et al., 2022) to focus on evaluating the inherent reasoning capabilities of LLMs, foregoing more advanced prompting techniques that could potentially improve performance but might obscure the models' true reasoning abilities. Finally, our current evaluation prioritizes reasoning correctness over efficiency, treating all solvable states equally regardless of the steps required to reach the solution. Future research could incorporate metrics like "steps to solution" to assess efficiency.

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.
- Yew Ken Chia, Guizhen Chen, Weiwen Xu, Anh Tuan Luu, Soujanya Poria, and Lidong Bing. 2024a. [Reasoning paths optimization: Learning to reason and explore from diverse paths](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 16763–16780, Miami, Florida, USA. Association for Computational Linguistics.
- Yew Ken Chia, Vernon Toh, Deepanway Ghosal, Lidong Bing, and Soujanya Poria. 2024b. [PuzzleVQA: Diagnosing multimodal reasoning challenges of language models with abstract visual patterns](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 16259–16273, Bangkok, Thailand. 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. 2024. [Everything of thoughts: Defying the law of penrose triangle for thought generation](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1638–1662, Bangkok, Thailand. Association for Computational Linguistics.
- Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. 2023. [Faith and fate: Limits of transformers on compositionality](#). In *Thirty-seventh Conference on Neural Information Processing Systems*.
- 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, Ziyang Wang, Hongrui Tang, Mengyue Yang, Kun Shao, David Henry Mguni, Yali Du, and Jun Wang. 2023. [ChessGPT: Bridging policy learning and language modeling](#). In *Thirty-seventh*

- Conference on Neural Information Processing Systems Datasets and Benchmarks Track.*
- Panagiotis Giadikiaroglou, Maria Lymperaioi, 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](#).
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. [Measuring massive multitask language understanding](#). In *International Conference on Learning Representations*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. [Measuring mathematical problem solving with the MATH dataset](#). In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.
- Kung-Hsiang Huang, Hou Pong Chan, May Fung, Haoyi Qiu, Mingyang Zhou, Shafiq Joty, Shih-Fu Chang, and Heng Ji. 2025. [From pixels to insights: A survey on automatic chart understanding in the era of large foundation models](#). *IEEE Trans. Knowl. Data Eng.*, 37(5):2550–2568.
- Kung-Hsiang Huang, Mingyang Zhou, Hou Pong Chan, Yi Fung, Zhenhailong Wang, Lingyu Zhang, Shih-Fu Chang, and Heng Ji. 2024. [Do lvlms understand charts? analyzing and correcting factual errors in chart captioning](#). In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 730–749. Association for Computational Linguistics.
- 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*.
- Xingxuan Li, Weiwen Xu, Ruochen Zhao, Fangkai Jiao, Shafiq Joty, and Lidong Bing. 2024a. [Can we further elicit reasoning in llms? critic-guided planning with retrieval-augmentation for solving challenging tasks](#). *arXiv preprint arXiv:2410.01428*.
- Yinghao Li, Haorui Wang, and Chao Zhang. 2024b. [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](#).
- 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, Jiayi 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. 2024. [GPQA: A graduate-level google-proof q&a benchmark](#). In *First Conference on Language Modeling*.
- 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. [Reflection: 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 Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. [Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning](#). In *The Thirteenth International Conference on Learning Representations*.
- 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. 2023. [Beyond the imitation game: Quantifying and extrapolating the capabilities of language models](#). *Transactions on Machine Learning Research*. Featured Certification.
- 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, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. [Chain of thought prompting elicits reasoning in large language models](#). In *Advances in Neural Information Processing Systems*.
- Larry Wos, Ross Overbeek, Ewing Lusk, and Jim Boyle. 1992. *Automated reasoning introduction and applications*. McGraw-Hill, Inc.
- Weiwenn Xu, Deng Cai, Zhisong Zhang, Wai Lam, and Shuming Shi. 2024. [Reasons to reject? aligning language models with judgments](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 12288–12304, Bangkok, Thailand. Association for Computational Linguistics.
- Weiwenn Xu, Yang Deng, Huihui Zhang, Deng Cai, and Wai Lam. 2021a. [Exploiting reasoning chains for multi-hop science question answering](#). In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1143–1156, Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Weiwenn Xu, Xin Li, Yang Deng, Wai Lam, and Lidong Bing. 2023. [PeerDA: Data augmentation via modeling peer relation for span identification tasks](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8681–8699, Toronto, Canada. Association for Computational Linguistics.
- Weiwenn Xu, Huihui Zhang, Deng Cai, and Wai Lam. 2021b. [Dynamic semantic graph construction and reasoning for explainable multi-hop science question answering](#). In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1044–1056, Online. Association for Computational Linguistics.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. [HotpotQA: A dataset for diverse, explainable multi-hop question answering](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

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](#). In *Advances in Neural Information Processing Systems*.

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

Table 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 a balance between solvable and unsolvable states for the remaining three puzzles.

Task	Questions	Solvable States	Unsolvable States
Sudoku	51	250	250
Graph Coloring	51	250	250
Game 24	98	250	250
Grid Puzzles	50	94	406

Table 8: Dataset Statistics

A.2 Prompt Templates

In Table 13, 14, 15, 16, 17, we show the state checking and state transition prompts for each puzzle.

A.3 Additional results on state checking precision, recall and F1 scores

Table 9 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 across 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.

A.4 Error types in state checking

Through human analysis, we categorise five common types of errors o1 made in state checking. We show one example for each type of error: Misinterpretation of Premises (Table 18), Exploration and Backtracking Issues (Table 19), Inconsistent Reasoning (Table 20), Conflicts Resolving Failure (Table 21), and False Premise (Table 22).

A.5 Error types in state transition

We automatically classify the state transition mistakes based on the next move. We show one example for each type of error models made in the

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

state transition: Multiple Moves (Table 23), Invalid Move (Table 24), Unsolvable Child (Table 25), Backtracking Failure (Table 26), Sibling (Table 27).

A.6 Training Details

We train our models using GRPO based on OpenR1³. 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 10.

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 GPUs (120 GB)

Table 10: Hyperparameter and training details.

A.7 Benefits of mixed-training across difficulty level

Table 11 analyses the impact of mixed training on performance across varying difficulty levels within the MATH dataset. Notably, the gains achieved through mixed training are most significant for levels 4 and 5 problems, which represent the most

³<https://github.com/huggingface/open-r1>

Difficulty Level	Math-only	Mixed	Difference (%)
Level 1	372	386	4%
Level 2	622	630	1%
Level 3	675	661	-2%
Level 4	540	588	9%
Level 5	348	389	12%

Table 11: The % improvement of correct samples from mixed training using the 1.5B model.

challenging instances. This suggests that the benefits of mixed training are amplified when tackling complex problems that necessitate more extensive reflection and iterative correction processes.

A.8 Quality analysis of model outputs with/without mixed training

Through quality analysis of 20 examples, we found more evidence of verification and self-correction in their thought process. Here we show two examples in Table 28.

In the first example, model 1 applied the formula directly without cross-checking its consistency with the given example of the third row. After training, model 2 verified the calculation against the provided example before reaching the answer.

In the second example, model 2 initially included an incorrect letter “C” but it caught this mistake and removed “C” before proceeding. However, model 1 skipped this step.

These examples suggest that puzzle training helps reinforce skills like constraint validation and error correction, which generalize to math reasoning.

A.9 Benefits of mixed-training across different reasoning tasks

In Table 12, we analyze the performance improvement across different subsets in MMLU. We organize the tasks into four categories following Hendrycks et al. (2021a). The most significant improvement is in STEM tasks, where accuracy increases by 10%. The STEM subset consists of multi-step reasoning problems that require reflection and correction in the reasoning process. We also observe some benefits (+3.8%) to the social sciences subset, as some subjects (eg, Econometrics) require multi-step reasoning. The humanities subset and other tasks, which are typically straightforward questions requiring fewer reasoning steps, show smaller improvements.

Category	MMLU-only	Mixed	Improvement (%)
Humanities	33.0	33.8	0.8
Social Sciences	45.4	49.2	3.8
STEM	48.7	58.7	10.0
Others	41.2	43.3	2.1

Table 12: The improvement in MMLU performance from mixed training using the 1.5B model across different task categories.

Sudoku 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)".

Sudoku 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:

$S(0) = [[0, 1, 0, 0, 7, 0, 8, 3, 0], [0, 2, 0, 0, 0, 0, 6, 0, 4], [5, 9, 0, 0, 0, 0, 0, 1, 0], [0, 0, 5, 0, 1, 9, 0, 0, 0], [1, 0, 0, 2, 6, 0, 0, 5, 3], [2, 4, 7, 0, 0, 8, 0, 0, 9], [0, 7, 0, 6, 9, 0, 0, 0, 1], [3, 0, 0, 0, 5, 0, 0, 7, 2], [0, 0, 0, 0, 0, 4, 3, 8, 0]]$

$L(0) = \text{Solvable}$

Two moves ago:

$S(i-2) = [[4, 1, 6, 9, 7, 2, 8, 3, 5], [7, 2, 3, 1, 8, 5, 6, 9, 4], [5, 9, 8, 3, 4, 6, 2, 1, 7], [6, 3, 5, 4, 1, 9, 7, 2, 8], [1, 8, 9, 2, 6, 7, 4, 5, 3], [2, 4, 7, 5, 3, 8, 1, 6, 9], [8, 7, 2, 6, 9, 3, 5, 0, 1], [3, 0, 0, 0, 5, 0, 0, 7, 2], [0, 0, 0, 0, 0, 4, 3, 8, 0]]$

$L(i-2) = \text{Solvable}$

One move ago:

$S(i-1) = [[4, 1, 6, 9, 7, 2, 8, 3, 5], [7, 2, 3, 1, 8, 5, 6, 9, 4], [5, 9, 8, 3, 4, 6, 2, 1, 7], [6, 3, 5, 4, 1, 9, 7, 2, 8], [1, 8, 9, 2, 6, 7, 4, 5, 3], [2, 4, 7, 5, 3, 8, 1, 6, 9], [8, 7, 2, 6, 9, 3, 5, 4, 1], [3, 0, 0, 0, 5, 0, 0, 7, 2], [0, 0, 0, 0, 0, 4, 3, 8, 0]]$

$L(i-1) = \text{Solvable}$

Current state:

$S(i) = [[4, 1, 6, 9, 7, 2, 8, 3, 5], [7, 2, 3, 1, 8, 5, 6, 9, 4], [5, 9, 8, 3, 4, 6, 2, 1, 7], [6, 3, 5, 4, 1, 9, 7, 2, 8], [1, 8, 9, 2, 6, 7, 4, 5, 3], [2, 4, 7, 5, 3, 8, 1, 6, 9], [8, 7, 2, 6, 9, 3, 5, 4, 1], [3, 6, 0, 0, 5, 0, 0, 7, 2], [0, 0, 0, 0, 0, 4, 3, 8, 0]]$

$L(i) = \text{Solvable}$

Explored next state:

$S(i+1) = [[4, 1, 6, 9, 7, 2, 8, 3, 5], [7, 2, 3, 1, 8, 5, 6, 9, 4], [5, 9, 8, 3, 4, 6, 2, 1, 7], [6, 3, 5, 4, 1, 9, 7, 2, 8], [1, 8, 9, 2, 6, 7, 4, 5, 3], [2, 4, 7, 5, 3, 8, 1, 6, 9], [8, 7, 2, 6, 9, 3, 5, 4, 1], [3, 6, 1, 0, 5, 0, 0, 7, 2], [0, 0, 0, 0, 0, 4, 3, 8, 0]]$

$L(i+1) = \text{Unsolvable}$

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 13: Prompt templates for state checking and state transition in Sudoku. Blue denotes question specific content.

Graph Coloring State Checking

You are given a graph represented as an adjacency list, where each index corresponds to a vertex, and the list at that index represents its adjacent vertices. You are also given the current coloring state of the graph in a list, where each index represents the corresponding vertex, and the number at that index represents its color (0 indicates an uncolored vertex).

Your task is to determine if this current state can lead to a valid coloring. Specifically, use lookahead techniques to determine if it's possible to color the remaining vertices such that no two adjacent vertices share the same color, using no more than 8 colors in total. Additionally, you are provided with a previously explored next state that has been proven to be uncolorable. Use this information to avoid revisiting this failed path and leverage it to make a more informed decision about the current state.

Graph adjacency list:

```
[[1, 4, 6, 7, 10, 11, 12, 14, 15, 16, 17, 18, 19, 21, 23], [0, 3, 5, 10, 12, 14, 15, 16, 17, 21, 22], [7, 10, 11, 12, 17, 18, 19, 21, 23], [1, 6, 7, 10, 11, 12, 14, 18, 21], [0, 6, 7, 10, 11, 12, 15, 20], [1, 6, 7, 8, 10, 11, 20], [0, 3, 4, 5, 9, 10, 11, 12, 14, 15, 16, 19, 20, 21, 22], [0, 2, 3, 4, 5, 10, 11, 13, 14, 18], [5, 9, 11, 13, 14, 16, 17, 19, 22, 23], [6, 8, 10, 14, 17, 23], [0, 1, 2, 3, 4, 5, 6, 7, 9, 13, 14, 16, 19, 22, 23], [0, 2, 3, 4, 5, 6, 7, 8, 12, 13, 14, 17, 19, 20], [0, 1, 2, 3, 4, 6, 11, 14, 15, 16, 18, 19, 21, 23], [7, 8, 10, 11, 14, 15, 18, 20, 22], [0, 1, 3, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 18, 19, 20, 21, 23], [0, 1, 4, 6, 12, 13, 14, 17, 18, 19, 20, 21, 23], [0, 1, 6, 8, 10, 12, 14, 17, 21, 22], [0, 1, 2, 8, 9, 11, 15, 16, 18], [0, 2, 3, 7, 12, 13, 14, 15, 17, 19, 21], [0, 2, 6, 8, 10, 11, 12, 14, 15, 18, 23], [4, 5, 6, 11, 13, 14, 15, 21, 22, 23], [0, 1, 2, 3, 6, 12, 14, 15, 16, 18, 20, 23], [1, 6, 8, 10, 13, 16, 20], [0, 2, 8, 9, 10, 12, 14, 15, 19, 20, 21]]
```

Current coloring state:

```
[1, 2, 1, 1, 2, 1, 3, 3, 2, 1, 4, 4, 5, 1, 6, 4, 7, 3, 0, 0, 0, 0, 0]
```

Explored next state that leads to an uncolorable path:

```
[1, 2, 1, 1, 2, 1, 3, 3, 2, 1, 4, 4, 5, 1, 6, 4, 7, 3, 8, 0, 0, 0, 0]
```

Let's think step by step, considering the failed state to avoid unnecessary exploration. Do not solve using programming.

Choose from (A) Colorable (B) Uncolorable. End your answer with "Answer: (A)" or "Answer: (B)".

Graph Coloring State Transition

You are given a graph represented as an adjacency list, where each index corresponds to a vertex, and the list at that index represents its adjacent vertices. You are also given a sequence of partial coloring states leading to the current coloring state $S(i)$. The coloring state is a list, where each index represents the corresponding vertex in the graph, and the number at that index represents its color (0 indicates an uncolored vertex). Alongside each state, its colorability status $L(*)$ is given.

Your task is to determine the next state by making exactly one move, ensuring progress toward a valid coloring with no more than 8 colors. A valid coloring requires that no two adjacent vertices share the same color.

Additionally, you are provided with a previously explored next state that has been proven to be uncolorable. Use this information to avoid revisiting this failed path.

A move is defined as either:

1. Coloring: Replacing a 0 in exactly one uncolored vertex with a value from 1 to 8.
2. Removing a color: Replacing a value in exactly one colored vertex with 0.

Graph adjacency list:

```
[[1, 4, 6, 7, 10, 11, 12, 14, 15, 16, 17, 18, 19, 21, 23], [0, 3, 5, 10, 12, 14, 15, 16, 17, 21, 22], [7, 10, 11, 12, 17, 18, 19, 21, 23], [1, 6, 7, 10, 11, 12, 14, 18, 21], [0, 6, 7, 10, 11, 12, 15, 20], [1, 6, 7, 8, 10, 11, 20], [0, 3, 4, 5, 9, 10, 11, 12, 14, 15, 16, 19, 20, 21, 22], [0, 2, 3, 4, 5, 10, 11, 13, 14, 18], [5, 9, 11, 13, 14, 16, 17, 19, 22, 23], [6, 8, 10, 14, 17, 23], [0, 1, 2, 3, 4, 5, 6, 7, 9, 13, 14, 16, 19, 22, 23], [0, 2, 3, 4, 5, 6, 7, 8, 12, 13, 14, 17, 19, 20], [0, 1, 2, 3, 4, 6, 11, 14, 15, 16, 18, 19, 21, 23], [7, 8, 10, 11, 14, 15, 18, 20, 22], [0, 1, 3, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 18, 19, 20, 21, 23], [0, 1, 4, 6, 12, 13, 14, 17, 18, 19, 20, 21, 23], [0, 1, 6, 8, 10, 12, 14, 17, 21, 22], [0, 1, 2, 8, 9, 11, 15, 16, 18], [0, 2, 3, 7, 12, 13, 14, 15, 17, 19, 21], [0, 2, 6, 8, 10, 11, 12, 14, 15, 18, 23], [4, 5, 6, 11, 13, 14, 15, 21, 22, 23], [0, 1, 2, 3, 6, 12, 14, 15, 16, 18, 20, 23], [1, 6, 8, 10, 13, 16, 20], [0, 2, 8, 9, 10, 12, 14, 15, 19, 20, 21]]
```

Two moves ago:

```
S(i-2) = [1, 2, 1, 1, 2, 1, 3, 3, 2, 1, 4, 4, 5, 1, 6, 4, 0, 0, 0, 0, 0, 0, 0]
```

$L(i-2)$ = Colorable

One move ago:

```
S(i-1) = [1, 2, 1, 1, 2, 1, 3, 3, 2, 1, 4, 4, 5, 1, 6, 4, 7, 0, 0, 0, 0, 0, 0]
```

$L(i-1)$ = Colorable

Current coloring state:

```
S(i) = [1, 2, 1, 1, 2, 1, 3, 3, 2, 1, 4, 4, 5, 1, 6, 4, 7, 3, 0, 0, 0, 0, 0]
```

$L(i)$ = Colorable

Explored next state:

```
S(i+1) = [1, 2, 1, 1, 2, 1, 3, 3, 2, 1, 4, 4, 5, 1, 6, 4, 7, 3, 8, 0, 0, 0, 0]
```

$L(i+1)$ = Uncolorable

Let's think step by step. Analyze the progress made so far and determine the immediate next move. End your answer with "Next state: {coloring}", where {coloring} is in the same python list format as the previous states.

Table 14: Prompt templates for state checking and state transition in Graph Coloring. Blue denotes question specific content.

Game of 24 State Checking

You are given four numbers and the current calculation state for the Game of 24.

Your task is to determine if this current state can lead to a solvable solution. Specifically, use lookahead techniques to determine if the remaining numbers can be combined using basic arithmetic operations (+ - * /) to reach exactly 24. You must use each number exactly once.

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.

Numbers:

`['5', '9', '12', '12']`

Current calculation state:

`['(5 + 9)', '12', '12']`

Explored next state that leads to an unsolvable path:

`['((5 + 9) + 12)', '12']`

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)".

Game of 24 State Transition

You are given an initial Game of 24 configuration $S(0)$, 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 solution requires using each of the four initial numbers exactly once, using only basic arithmetic operations (+ - * /), and ultimately evaluating to 24.

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. Applying an operation: Combining two expressions using a basic arithmetic operation (+ - * /), reducing the number of expressions by 1.
2. Reverting an operation: Removing the last operation applied to the expressions, increasing the number of expressions by 1.

Initial configuration:

$S(0) = ['5', '9', '12', '12']$

$L(0) = \text{Solvable}$

Current state:

$S(i) = ['(5 + 9)', '12', '12']$

$L(i) = \text{Solvable}$

Explored next state:

$S(i+1) = ['((5 + 9) + 12)', '12']$

$L(i+1) = \text{Unsolvable}$

Let's think step by step. Analyze the progress made so far and determine the immediate next move. End your answer with "Next state: {expressions}", where {expressions} is in the same python list format as the previous states.

Table 15: Prompt templates for state checking and state transition in Game of 24. Blue denotes question specific content.

Grid Puzzle State Checking

You are given a partially filled logic grid puzzle represented as a table, where each column corresponds to a specific category, and each row represents attributes of a distinct entry. Empty cells are represented as the empty string ('').

Your task is to determine if this current state can lead to a solvable solution. Specifically, use lookahead techniques to determine if the current configuration can lead to a valid solution under standard logic puzzle constraints (each option in every category must only appear once and adhere to the given clues).

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.

Question:

Help Donna make sense of Dr. Finklestein's appointment list for today. Using only the clues below, match the times to the options from names, ailments, and insurers. Remember, as with all grid-based logic puzzles, no option in any category will ever be used more than once.

Categories:

times : 9, 10, 11, 12.

names : Billy, Guy, Paul, Terry.

ailments : back pain, hip pain, shingles, vertigo.

insurers : Ambercare, HealthCo, Lifealign, Triflex.

Clues:

1. The person with Lifealign insurance has an appointment sometime before the patient suffering from shingles.
2. The patient with the 12 noon appointment is either Terry or the patient with Ambercare insurance.
3. The patient suffering from back pain has an appointment 2 hours before Guy.
4. The person with Ambercare insurance has an appointment sometime after the person suffering from vertigo.
5. Neither Billy nor the person suffering from shingles is the person with Lifealign insurance.
6. The person with the 9:00am appointment is either Paul or the patient suffering from back pain.
7. The patient with the 10:00am appointment has Triflex insurance.
8. Of the patient suffering from vertigo and the person with Ambercare insurance, one has the 11:00am appointment and the other is Billy.

Initial state:

S(0) = [['times', 'names', 'ailments', 'insurers'], [9, "", "", ""], [10, "", "", ""], [11, "", "", ""], [12, "", "", "]]

State 1 (Current state):

Clue applied: 8

S(1) = [['times', 'names', 'ailments', 'insurers'], [9, "", "", ""], [10, "", "", ""], [11, "", 'vertigo', ""], [12, 'Billy', "", 'Ambercare']]

Explored next state that leads to an unsolvable path:

Clue applied: 2

S(2) = [['times', 'names', 'ailments', 'insurers'], [9, "", "", ""], [10, 'Terry', "", ""], [11, "", 'vertigo', ""], [12, 'Billy', "", 'Ambercare']]

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)".

Table 16: Prompt template for state checking in Grid Puzzles. Blue denotes question specific content.

Grid Puzzle State Transition

You are given a logic grid puzzle represented as a table, where each column corresponds to a specific category, and each row represents attributes of a distinct entry. Empty cells are represented as the empty string (''). You are also given a sequence of progressive states from the initial state $S(0)$ to the current state $S(n)$. Alongside each state, its solvability status $L(*)$ is provided. Your task is to determine the next state by making exactly one move, ensuring progress toward a valid solution. A valid solution requires that each option in every category appears only once, strictly following the given clues.

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. Applying a clue: Filling the table with the values indicated by that clue, as long as it does not conflict with any existing clues or placed options.
2. Reverting a clue: Removing the last operation applied to the table.

Question:

Help Donna make sense of Dr. Finklestein's appointment list for today. Using only the clues below, match the times to the options from names, ailments, and insurers. Remember, as with all grid-based logic puzzles, no option in any category will ever be used more than once.

Categories:

times : 9, 10, 11, 12.

names : Billy, Guy, Paul, Terry.

ailments : back pain, hip pain, shingles, vertigo.

insurers : Ambercare, HealthCo, Lifealign, Triflex.

Clues:

1. The person with Lifealign insurance has an appointment sometime before the patient suffering from shingles.
2. The patient with the 12 noon appointment is either Terry or the patient with Ambercare insurance.
3. The patient suffering from back pain has an appointment 2 hours before Guy.
4. The person with Ambercare insurance has an appointment sometime after the person suffering from vertigo.
5. Neither Billy nor the person suffering from shingles is the person with Lifealign insurance.
6. The person with the 9:00am appointment is either Paul or the patient suffering from back pain.
7. The patient with the 10:00am appointment has Triflex insurance.
8. Of the patient suffering from vertigo and the person with Ambercare insurance, one has the 11:00am appointment and the other is Billy.

Initial state:

$S(0) = [['times', 'names', 'ailments', 'insurers'], [9, "", "", ""], [10, "", "", ""], [11, "", "", ""], [12, "", "", "]]$

$L(0) = \text{Solvable}$

State 1 (Current state):

Clue applied: 8

$S(1) = [['times', 'names', 'ailments', 'insurers'], [9, "", "", ""], [10, "", "", ""], [11, "", 'vertigo', ""], [12, 'Billy', "", 'Ambercare']]$

$L(1) = \text{Solvable}$

Explored next state:

Clue applied: 2

$S(2) = [['times', 'names', 'ailments', 'insurers'], [9, "", "", ""], [10, 'Terry', "", ""], [11, "", 'vertigo', ""], [12, 'Billy', "", 'Ambercare']]$

$L(2) = \text{Unsolvable}$

Let's think step by step. Analyze the progress made so far and determine the immediate next move. End your answer with "Next state: {table}", where {table} is in the same python list format as the previous states.

Table 17: Prompt template for state transition in Grid Puzzles. Blue denotes question specific content.

A question from Grid Puzzle State Checking

{Grid Puzzle State Checking instructions}

Categories:

earnings : 2, 5, 8, 11.

movies : Amerigo, Easy to Love, Tippecanoe, Zephyr Breeze.

directors : Hick Harmon, Micah Moreno, Nan Norman, Odette Olson.

Clues:

1. Amerigo made 6 million less than the movie directed by Hick Harmon.
2. Zephyr Breeze made 6 million more than Amerigo.
3. The movie directed by Micah Moreno made somewhat more than Tippecanoe.
4. Of the movie directed by Nan Norman and Easy to Love, one made \$11 million and the other made \$2 million.

Initial state:

S(0) = [['earnings', 'movies', 'directors'], [2, "", ""], [5, "", ""], [8, "", ""], [11, "", ""]]

State 1 (Current state):

Clue applied: 4

S(1) = [['earnings', 'movies', 'directors'], [2, "", 'Nan Norman'], [5, "", ""], [8, "", ""], [11, 'Easy to Love', ""]]

Explored next state that leads to an unsolvable path:

Clue applied: 1

S(2) = [['earnings', 'movies', 'directors'], [2, "", 'Nan Norman'], [5, 'Amerigo', ""], [8, "", ""], [11, 'Easy to Love', 'Hick Harmon']]

Output from o1 (Misinterpretation of Premises)

To determine whether the current state S(1) can lead to a valid solution, we need to analyze the clues carefully while considering the information from the failed state S(2).

Given:

- **Current State S(1):**

- \$2 million: Director Nan Norman
- \$11 million: Movie Easy to Love

- **Failed State S(2):**

- Applied Clue 1 leading to an unsolvable path.

Analysis:

1. **Clue 4 Applied in S(1):**

- "Of the movie directed by Nan Norman and Easy to Love, one made \$11 million and the other made \$2 million."
- In S(1), Nan Norman is at \$2 million, and Easy to Love is at \$11 million.

2. **Possible Earnings Assignments Based on Clues 1 & 2:**

- **Clue 1:** "Amerigo made \$6 million less than the movie directed by Hick Harmon."
- **Clue 2:** "Zephyr Breeze made \$6 million more than Amerigo."

- This implies:

- **Option A:**

- Amerigo at \$5 million, Hick Harmon's movie at \$11 million.
- Zephyr Breeze would need to be at \$11 million (which conflicts with Easy to Love at \$11 million).

- **Option B:**

- Amerigo at \$2 million, Hick Harmon's movie at \$8 million.
- Zephyr Breeze would need to be at \$8 million (possible).

3. **Conflict with Clue 4:**

- **Assigning Amerigo to \$2 million places it with Nan Norman at the same earnings, which violates the rule that each earnings amount is unique per movie and director in this puzzle.**

4. **Clue 3 Introduces Further Conflicts:**

- "The movie directed by Micah Moreno made somewhat more than Tippecanoe."
- With limited earnings slots, we cannot assign Micah Moreno and Tippecanoe without conflicts while satisfying previous assignments and clues.

5. **Conclusion from Analysis:**

- All possible assignments lead to conflicts or violate the puzzle's constraints.
- The failed state S(2) reinforces that applying Clue 1 after Clue 4 leads to an unsolvable path.
- No alternative assignments in S(1) avoid these conflicts.

Therefore, considering the current state and the previous failed state, we conclude that the puzzle cannot be solved from S(1).

Table 18: An example of Misinterpretation of Premises during state checking in Grid Puzzles. The model wrongly concludes that **assigning Amerigo to \$2 million alongside Nan Norman violates uniqueness. This is a misinterpretation between the movie and director categories.**

A question from Grid Puzzle State Checking

{Grid Puzzle State Checking instructions }

Categories:

dollars: 250, 275, 300, 325, 350, 375.

woodworkers : Bonita, Ellen, Josephine, Kendra, Olivia, Patsy.

woods : ash, mahogany, oak, poplar, rosewood, walnut.

pieces : barstool, bookshelf, coffee table, end table, footstool, hutch.

Clues:

1. The \$325 piece is the barstool.
2. Of the piece made of ash and the barstool, one was crafted by Bonita and the other costs \$250.
3. The end table costs 25 dollars more than Ellen's item.
4. Ellen's piece costs 50 dollars more than the hutch.
5. The bookshelf wasn't made of oak.
6. Kendra's item wasn't made of rosewood.
7. Neither the coffee table nor the item made of oak is Kendra's piece.
8. Josephine's item is either the item made of walnut or the end table.
9. Of Bonita's piece and the bookshelf, one was made of poplar and the other costs \$275.
10. The piece made of walnut costs 75 dollars more than Patsy's piece.
11. Neither the piece made of oak nor the item made of walnut is the coffee table.

{Previous states skipped here... Previous clues applied: 2, 9 }

State 3 (Current state):

Clue applied: 1

S(3) = [['dollars', 'woodworkers', 'woods', 'pieces'], [250, '', 'ash', ''], [275, '', '', 'bookshelf'], [300, '', '', ''], [325, 'Bonita', 'poplar', 'barstool'], [350, '', '', ''], [375, '', '', '']]

Explored next state that leads to an unsolvable path:

Clue applied: 3

S(4) = [['dollars', 'woodworkers', 'woods', 'pieces'], [250, '', 'ash', ''], [275, 'Ellen', '', 'bookshelf'], [300, '', '', 'end table'], [325, 'Bonita', 'poplar', 'barstool'], [350, '', '', ''], [375, '', '', '']]

Clue analysis from o1 (Exploration and Backtracking Issues)

1. **Clue 3 & 4 (Ellen's Item and the End Table):**
 - Ellen's item costs \$50 more than the hutch. The end table costs \$25 more than Ellen's item.
 - Assume Ellen's item is at \$350. Hutch would be at \$300. End table would be at \$375.
 - These prices are available, so we assign:
 - **\$350 row**: Ellen's item
 - **\$375 row**: End table
 - **\$300 row**: Hutch
 2. **Clue 10 (Walnut and Patsy's Item):**
 - The piece made of walnut costs \$75 more than Patsy's piece. **Potential pairs are (\$300, \$375), (\$275, \$350), or (\$250, \$325).**
 - Considering previous assignments: **Assign walnut to \$375 (end table). Patsy's item would be at \$300 (hutch).**
 - Assign:
 - **\$375 row**: Walnut, end table
 - **\$300 row**: Patsy's item, hutch
 3. **Clue 8 (Josephine's Item):**
 - Josephine's item is either the item made of walnut or the end table.
 - Since the end table is at \$375 and made of walnut, Josephine must have the end table.
 - Assign:
 - **\$375 row**: Josephine, walnut, end table
 4. **Clue 11 (Coffee Table Not Made of Oak or Walnut):**
 - Since walnut is already assigned to the end table, the coffee table can't be made of oak or walnut.
 - Remaining woods for the coffee table: Mahogany or Rosewood.
 5. **Clues 6 & 7 (Kendra's Item):**
 - Kendra's item wasn't made of rosewood and isn't the coffee table or made of oak. The only wood left for Kendra that's not rosewood or oak is mahogany. Possible pieces are footstool or the piece at \$250 with ash (already assigned to ash). There's no available price slot for Kendra's item without causing conflicts.
-

Table 19: An example of Exploration and Backtracking Issues in Grid Puzzles. **The model correctly lists all possible assignments, but fails to backtrack and explore alternative assignments when conflicts arise.**

A question from Grid Puzzle State Checking

{Grid Puzzle State Checking instructions}

Categories:

orders : 1, 2, 3, 4, 5, 6.

names : Howard, Katrina, Melissa, Nick, Peter, Shari.

items : balls, batons, bowling pins, chainsaws, flashlights, mobile phones.

towns : Carbon, Dubuque, Menlo, Nice, Quasqueton, Selma.

Clues:

1. The performer from Quasqueton didn't use balls.
2. The person who went sixth wasn't from Carbon.
3. The person from Nice didn't use mobile phones.
4. Howard went fourth.
5. The person who went first didn't use mobile phones.
6. The juggler from Menlo was either the juggler who used chainsaws or Nick.
7. Of the performer who went sixth and the juggler who went fifth, one used flashlights and the other was from Menlo.
8. Melissa was either the juggler who used chainsaws or the performer from Selma.
9. Nick performed 3 spots before the juggler from Nice.
10. The juggler from Menlo performed 2 spots after Melissa.
11. Shari didn't go fifth.

Initial state:

S(0) = [['orders', 'names', 'items', 'towns'], [1, "", "", ""], [2, "", "", ""], [3, "", "", ""], [4, "", "", ""], [5, "", "", ""], [6, "", "", ""]]

State 1 (Current state):

Clue applied: 7

S(1) = [['orders', 'names', 'items', 'towns'], [1, "", "", ""], [2, "", "", ""], [3, "", "", ""], [4, "", "", ""], [5, "", 'flashlights', ""], [6, "", 'Menlo']]

Explored next state that leads to an unsolvable path:

Clue applied: 10

S(2) = [['orders', 'names', 'items', 'towns'], [1, "", "", ""], [2, "", "", ""], [3, "", "", ""], [4, 'Melissa', "", ""], [5, "", 'flashlights', ""], [6, "", 'Menlo']]

Output from o1 (Inconsistent Reasoning)

To determine if the current state S(1) can lead to a solvable solution, let's analyze the clues and the given failed path.

****Given Failed Path:**** In S(2), assigning "Melissa" to order 4 leads to a contradiction because Clue 4 states "Howard went fourth," so "Melissa" cannot be in order 4. This suggests that assigning "Menlo" to order 6 (as in S(1)) and then applying Clue 10 leads to an unsolvable situation.

****Alternative Approach:****

1. ****Re-evaluate Clue 7:**** Clue 7 says, "Of the performer who went sixth and the juggler who went fifth, one used flashlights and the other was from Menlo." In S(1), we assigned "flashlights" to order 5 and "Menlo" to order 6. However, we could instead assign "Menlo" to order 5 and "flashlights" to order 6.

2. ****Apply Clue 10 with New Assignments:**** With "Menlo" at order 5, Clue 10 ("The juggler from Menlo performed 2 spots after Melissa.") means "Melissa" performed at order 3, which is possible since that spot is unassigned.

3. ****Check for Consistency:**** This new assignment doesn't conflict with any other clues. "Howard" is still at order 4 (Clue 4), and "Melissa" is at order 3.

****Conclusion:**** By adjusting our application of Clue 7 and avoiding the failed path, we find that the puzzle can still be solved from the current state S(1). The previously explored unsolvable path helps us make a more informed decision to choose an alternative assignment that leads to a solution.

Table 20: An example of Inconsistent Reasoning during state checking in Grid Puzzles. The model inconsistently suggests changing S(1)'s assignments to solve it, then concludes that the original S(1) is solvable.

A question from Grid Puzzle State Checking

{Grid Puzzle State Checking instructions}

Categories:

days : 3, 4, 5, 6, 7.

requests : clown, petting zoo, photo booth, rock band, superhero.

families : Benton, Hayden, Ingram, Stanley, Zimmerman.

Clues:

1. The party on October 5 is for the Ingrams.
2. The booking that requested the clown is sometime after the Ingram family booking.
3. The booking on October 4 is either the party that requested the petting zoo or the Hayden family booking.
4. The five events are the booking on October 7, the party that requested the rock band, the event that requested the superhero, the Ingram family party and the Benton family party.
5. The booking that requested the photo booth is either the event on October 6 or the event on October 3.
6. The Ingram family booking is 1 day before the booking that requested the superhero.
7. The party that requested the superhero is for the Zimmermans.

{Previous states skipped here... Previous clues applied: 5, 3}

State 3 (Current state):

Clue applied: 1

S(3) = [['days', 'requests', 'families'], [3, 'photo booth', ''], [4, 'petting zoo', ''], [5, '', 'Ingram'], [6, '', ''], [7, '', 'Hayden']]

Explored next state that leads to an unsolvable path:

Clue applied: 4

S(4) = [['days', 'requests', 'families'], [3, 'photo booth', 'Benton'], [4, 'petting zoo', ''], [5, '', 'Ingram'], [6, 'rock band', ''], [7, '', 'Hayden']]

Clue analysis from o1 (Conflicts Resolving Failure)

1. **Clue 7**: The party that requested the superhero is for the Zimmermans.
 - We don't have the 'superhero' request assigned yet.
 2. **Clue 6**: The Ingram family booking is 1 day before the booking that requested the superhero.
 - Since Ingram is on October 5, the 'superhero' request must be on October 6.
 - From Clue 7, the 'superhero' request is for the Zimmermans.
 - So, October 6: Requests 'superhero', Family 'Zimmerman'.
 3. **Clue 2**: The booking that requested the clown is sometime after the Ingram family booking.
 - After October 5, so it must be on October 7. So, October 7: Requests 'clown', Family 'Ingram'.
 4. **Clue 4**: The five events are:
 - The booking on October 7 (we have 'clown' request here).
 - The party that requested the rock band (not assigned yet).
 - The event that requested the superhero (October 6).
 - The Ingram family party (October 5).
 - The Benton family party (not assigned yet).
 - Since the 'rock band' request and Benton family are among these five events, they must be assigned to the remaining days.
 5. **Assigning Benton Family**:
 - Cannot be on October 5 or 6 (already assigned to Ingram and Zimmerman).
 - Cannot be on October 7 (it's the 'clown' request).
 - So, Benton family must be on October 3. So, October 3: Requests 'photo booth', Family 'Benton'.
 6. **Assigning 'rock band' Request**:
 - Must be one of the five events.
 - Remaining day is October 5 (occupied by Ingram) and October 4.
 - October 4 is 'petting zoo' (from Clue 3).
 - **So, 'rock band' request must be on October 5 with Ingram family. So, October 5: Requests 'rock band', Family 'Ingram'.**
 7. **Assigning Remaining Family (Hayden and Stanley)**:
 - From Clue 3, since Hayden is not on October 4, Hayden must be on October 7.
 - So, October 7: Requests 'clown', Family 'Hayden'.
 - Stanley family is then assigned to October 4.
 - So, October 4: Requests 'petting zoo', Family 'Stanley'.
- Final Assignment**:
- October 3: Requests 'photo booth', Family 'Benton'
 - October 4: Requests 'petting zoo', Family 'Stanley'
 - October 5: Requests 'rock band', Family 'Ingram'
 - October 6: Requests 'superhero', Family 'Zimmerman'
 - October 7: Requests 'clown', Family 'Hayden'
-

Table 21: An example of Conflicts Resolving Failure during state checking in Grid Puzzles. **The model assigns 'Ingram' and 'rock band' on the same row, which conflicts with clue 4.**

A question from Grid Puzzle State Checking

{Grid Puzzle State Checking instructions}

Categories:

dollars: 250, 275, 300, 325, 350, 375.

woodworkers : Bonita, Ellen, Josephine, Kendra, Olivia, Patsy.

woods : ash, mahogany, oak, poplar, rosewood, walnut.

pieces : barstool, bookshelf, coffee table, end table, footstool, hutch.

Clues:

1. The \$325 piece is the barstool.
2. Of the piece made of ash and the barstool, one was crafted by Bonita and the other costs \$250.
3. The end table costs 25 dollars more than Ellen's item.
4. Ellen's piece costs 50 dollars more than the hutch.
5. The bookshelf wasn't made of oak.
6. Kendra's item wasn't made of rosewood.
7. Neither the coffee table nor the item made of oak is Kendra's piece.
8. Josephine's item is either the item made of walnut or the end table.
9. Of Bonita's piece and the bookshelf, one was made of poplar and the other costs \$275.
10. The piece made of walnut costs 75 dollars more than Patsy's piece.
11. Neither the piece made of oak nor the item made of walnut is the coffee table.

{Previous states skipped here... Previous clues applied: 2, 9, 1, 3, 4, 8}

State 7 (Current state):

Clue applied: 10

S(7) = [['dollars', 'woodworkers', 'woods', 'pieces'], [250, '', 'ash', ''], [275, 'Patsy', '', 'bookshelf'], [300, '', '', 'hutch'], [325, 'Bonita', 'poplar', 'barstool'], [350, 'Ellen', 'walnut', ''], [375, 'Josephine', '', 'end table']]

Explored next state that leads to an unsolvable path:

Clue applied: 11

S(8) = [['dollars', 'woodworkers', 'woods', 'pieces'], [250, '', 'ash', 'coffee table'], [275, 'Patsy', 'oak', 'bookshelf'], [300, '', '', 'hutch'], [325, 'Bonita', 'poplar', 'barstool'], [350, 'Ellen', 'walnut', 'footstool'], [375, 'Josephine', '', 'end table']]

Clue analysis from o1 (False Premise)

1. **Clue 1** is satisfied: The **\$325** piece is the **Barstool**.
2. **Clue 2** is satisfied with current assignments: **Ash** piece is at **\$250** (not crafted by **Bonita**). **Barstool** (crafted by **Bonita**) is at **\$325**.
3. **Clue 3** is satisfied: **End Table** is at **\$375**. **Ellen's** item is at **\$350**. ($375 = 350 + 25$).
4. **Clue 4** is satisfied: **Ellen's** piece is at **\$350**. **Hutch** is at **\$300**. ($350 = 300 + 50$).
5. **Clue 5**: **Bookshelf** (at **\$275**) wasn't made of **Oak**.
6. **Clue 6**: **Kendra's** item wasn't made of **Rosewood**.
7. **Clue 7**: **Kendra's** item is neither the **Coffee Table** nor made of **Oak**.
8. **Clue 8** is satisfied: **Josephine's** item is the **End Table**.
9. **Clue 9** is satisfied: **Bonita's** piece (**Barstool**) is at **\$325** and made of **Poplar**. **Bookshelf** is at **\$275**. One is made of **Poplar**, the other costs **\$275**.
10. **Clue 10** is satisfied: **Walnut** piece is at **\$350** (**Ellen's** item). **Patsy's** piece is at **\$275**. ($350 = 275 + 75$).
11. **Clue 11**: **Coffee Table** is neither made of **Oak** nor **Walnut**.

Attempting to Assign Remaining Options:

- **Kendra** cannot be assigned to **\$250** (since that's the **Ash** piece).
- **Kendra** cannot be assigned to **\$275** (already assigned to **Patsy**).
- **Kendra** cannot be assigned to **\$300** (piece is **Hutch**, but **Kendra's** piece isn't specified yet).
- **Kendra** cannot be at **\$350** (already assigned to **Ellen**).
- **Kendra** cannot be at **\$375** (already assigned to **Josephine**).

This leaves no valid position for **Kendra**, given the constraints from the clues.

Table 22: An example of False Premise during state checking in Grid Puzzles. The question did not mention Kendra cannot be assigned to the Ash piece. This is a nonexistent constraint.

A question from Sudoku 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 3×3 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:

$S(0) = [[5, 3, 0, 0, 2, 7, 0, 0, 0], [0, 6, 7, 0, 0, 0, 0, 1], [4, 8, 0, 3, 0, 0, 0, 0, 9], [0, 0, 0, 8, 0, 5, 4, 0, 0], [8, 0, 2, 0, 9, 1, 0, 3, 0], [3, 1, 0, 4, 0, 2, 0, 7, 0], [9, 0, 0, 6, 0, 0, 0, 4, 5], [0, 0, 0, 0, 1, 4, 0, 0, 7], [0, 0, 0, 0, 8, 0, 2, 6, 0]]$

$L(0) = \text{Solvable}$

Two moves ago:

$S(i-2) = [[5, 3, 9, 1, 2, 7, 6, 8, 4], [2, 6, 7, 9, 4, 8, 3, 5, 1], [4, 8, 1, 3, 5, 6, 7, 2, 9], [7, 9, 6, 8, 3, 5, 4, 1, 2], [8, 4, 2, 7, 9, 1, 5, 3, 6], [3, 1, 5, 4, 6, 2, 9, 7, 0], [9, 0, 0, 6, 0, 0, 0, 4, 5], [0, 0, 0, 0, 1, 4, 0, 0, 7], [0, 0, 0, 0, 8, 0, 2, 6, 0]]$

$L(i-2) = \text{Solvable}$

One move ago:

$S(i-1) = [[5, 3, 9, 1, 2, 7, 6, 8, 4], [2, 6, 7, 9, 4, 8, 3, 5, 1], [4, 8, 1, 3, 5, 6, 7, 2, 9], [7, 9, 6, 8, 3, 5, 4, 1, 2], [8, 4, 2, 7, 9, 1, 5, 3, 6], [3, 1, 5, 4, 6, 2, 9, 7, 8], [9, 0, 0, 6, 0, 0, 0, 4, 5], [0, 0, 0, 0, 1, 4, 0, 0, 7], [0, 0, 0, 0, 8, 0, 2, 6, 0]]$

$L(i-1) = \text{Solvable}$

Current state:

$S(i) = [5, 3, 9, 1, 2, 7, 6, 8, 4], [2, 6, 7, 9, 4, 8, 3, 5, 1], [4, 8, 1, 3, 5, 6, 7, 2, 9], [7, 9, 6, 8, 3, 5, 4, 1, 2], [8, 4, 2, 7, 9, 1, 5, 3, 6], [3, 1, 5, 4, 6, 2, 9, 7, 8], [9, 2, 0, 6, 0, 0, 0, 4, 5], [0, 0, 0, 0, 1, 4, 0, 0, 7], [0, 0, 0, 0, 8, 0, 2, 6, 0]]$

$L(i) = \text{Solvable}$

Explored next state:

$S(i+1) = [[5, 3, 9, 1, 2, 7, 6, 8, 4], [2, 6, 7, 9, 4, 8, 3, 5, 1], [4, 8, 1, 3, 5, 6, 7, 2, 9], [7, 9, 6, 8, 3, 5, 4, 1, 2], [8, 4, 2, 7, 9, 1, 5, 3, 6], [3, 1, 5, 4, 6, 2, 9, 7, 8], [9, 2, 3, 6, 0, 0, 0, 4, 5], [0, 0, 0, 0, 1, 4, 0, 0, 7], [0, 0, 0, 0, 8, 0, 2, 6, 0]]$

$L(i+1) = \text{Unsolvable}$

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.

Output from o1 (Multiple Moves)

To determine the next move from the current state $S(i)$, we need to make an informed decision that avoids the previously explored unsolvable path and makes progress toward solving the Sudoku puzzle.

****Analysis of Previous Moves:****

- ****From $S(i-2)$ to $S(i-1)$:** Filled cell at (Row 6, Col 9) with ****8****.
- ****From $S(i-1)$ to $S(i)$:** Filled cell at (Row 7, Col 2) with ****2****.
- ****Explored and found unsolvable $S(i+1)$:** Filled cell at (Row 7, Col 3) with ****3****.

****Current Empty Cells in $S(i)$ and Potential Moves:****

- ****Row 7:** Cells at positions (Row 7, Col 3), (Row 7, Col 5), (Row 7, Col 6), (Row 7, Col 7)
- ****Row 8 and Row 9:** Multiple empty cells

We need to choose an empty cell that, when filled, adheres to Sudoku rules.

****Choosing Cell at (Row 7, Col 5):****

- ****Cell Position:** (Row 7, Col 5)
- ****Current Value:** ****0****

****Possible Numbers for This Cell:****

1. ****Numbers Already in Row 7:****
 - ****Existing Numbers:** ****9, 2, 6, 4, 5****
2. ****Numbers Already in Column 5:****
 - ****Existing Numbers:** ****2, 4, 5, 6, 8, 9, 1, **7 (after previous moves)****
3. ****Numbers Already in the 3×3 Block:****
 - ****Existing Numbers:** ****6, 1, 4, 8, **7 (after previous moves)****

****Available Numbers:** The number ****3**** is not present in the row, column, or block.

****Next Move:****

- ****Action:** Fill cell at (Row 7, Col 5) with ****3****.
 - ****Reasoning:** ****3**** is the only number that doesn't conflict with existing numbers in the row, column, or block.
- **Next State:****

Next state: $[[5, 3, 9, 1, 2, 7, 6, 8, 4], [2, 6, 7, 9, 4, 8, 3, 5, 1], [4, 8, 1, 3, 5, 6, 7, 2, 9], [7, 9, 6, 8, 3, 5, 4, 1, 2], [8, 4, 2, 7, 9, 1, 5, 3, 6], [3, 1, 5, 4, 6, 2, 9, 7, 8], [9, 2, 0, 6, 7, 3, 0, 4, 5], [0, 0, 0, 0, 1, 4, 0, 0, 7], [0, 0, 0, 0, 8, 0, 2, 6, 0]]$

Table 23: An example of making multiple moves during state transition in Sudoku. The model fills two numbers, 7 and 3, into the current state.

A question from Sudoku 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 3×3 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:

$S(0) = [[0, 0, 0, 3, 6, 1, 0, 9, 0], [5, 0, 9, 0, 0, 8, 0, 0, 0], [0, 1, 8, 0, 0, 0, 0, 2, 0], [8, 0, 4, 6, 0, 0, 3, 0, 0], [7, 3, 0, 2, 0, 9, 0, 0, 1], [0, 0, 0, 7, 0, 6, 8, 0], [0, 0, 2, 0, 0, 0, 0, 6, 4], [6, 0, 5, 1, 4, 0, 0, 0, 8], [0, 9, 0, 0, 5, 0, 2, 0, 7]]$

$L(0) = \text{Solvable}$

Two moves ago:

$S(i-2) = [[2, 4, 7, 3, 6, 1, 8, 9, 5], [5, 6, 9, 7, 2, 8, 1, 4, 3], [3, 1, 8, 5, 9, 4, 7, 2, 6], [8, 2, 4, 6, 1, 5, 3, 7, 9], [7, 3, 6, 2, 8, 9, 0, 0, 1], [0, 0, 0, 7, 0, 6, 8, 0], [0, 0, 2, 0, 0, 0, 0, 6, 4], [6, 0, 5, 1, 4, 0, 0, 0, 8], [0, 9, 0, 0, 5, 0, 2, 0, 7]]$

$L(i-2) = \text{Solvable}$

One move ago:

$S(i-1) = [[2, 4, 7, 3, 6, 1, 8, 9, 5], [5, 6, 9, 7, 2, 8, 1, 4, 3], [3, 1, 8, 5, 9, 4, 7, 2, 6], [8, 2, 4, 6, 1, 5, 3, 7, 9], [7, 3, 6, 2, 8, 9, 4, 0, 1], [0, 0, 0, 7, 0, 6, 8, 0], [0, 0, 2, 0, 0, 0, 0, 6, 4], [6, 0, 5, 1, 4, 0, 0, 0, 8], [0, 9, 0, 0, 5, 0, 2, 0, 7]]$

$L(i-1) = \text{Solvable}$

Current state:

$S(i) = [[2, 4, 7, 3, 6, 1, 8, 9, 5], [5, 6, 9, 7, 2, 8, 1, 4, 3], [3, 1, 8, 5, 9, 4, 7, 2, 6], [8, 2, 4, 6, 1, 5, 3, 7, 9], [7, 3, 6, 2, 8, 9, 4, 5, 1], [0, 0, 0, 7, 0, 6, 8, 0], [0, 0, 2, 0, 0, 0, 0, 6, 4], [6, 0, 5, 1, 4, 0, 0, 0, 8], [0, 9, 0, 0, 5, 0, 2, 0, 7]]$

$L(i) = \text{Solvable}$

Explored next state:

$S(i+1) = [[2, 4, 7, 3, 6, 1, 8, 9, 5], [5, 6, 9, 7, 2, 8, 1, 4, 3], [3, 1, 8, 5, 9, 4, 7, 2, 6], [8, 2, 4, 6, 1, 5, 3, 7, 9], [7, 3, 6, 2, 8, 9, 4, 5, 1], [1, 0, 0, 0, 7, 0, 6, 8, 0], [0, 0, 2, 0, 0, 0, 0, 6, 4], [6, 0, 5, 1, 4, 0, 0, 0, 8], [0, 9, 0, 0, 5, 0, 2, 0, 7]]$

$L(i+1) = \text{Unsolvable}$

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.

Output from o1 (Invalid Move)

To solve this problem, we need to find the next valid move from the current Sudoku state $S(i)$, ensuring progress toward a solution and avoiding previously failed paths. Here's the step-by-step analysis:

****Step 1: Analyze the Previous Moves****

- ****From $S(i-2)$ to $S(i-1)$ **:** We filled the number ****4**** at position (row 4, col 6).
- ****From $S(i-1)$ to $S(i)$ **:** We filled the number ****5**** at position (row 4, col 7).
- ****Attempted Move in $S(i+1)$ **:** Filled the number ****1**** at position (row 5, col 0), which led to an unsolvable state.

****Step 2: Identify Empty Cells in $S(i)$ ****

List all positions with zeros (empty cells):

- ****Row 5**:** (5,0), (5,1), (5,2), (5,3), (5,5), (5,8)
- ****Row 6**:** (6,0), (6,1), (6,3), (6,4), (6,5), (6,6)
- ****Row 7**:** (7,1), (7,5), (7,6), (7,7)
- ****Row 8**:** (8,0), (8,2), (8,3), (8,5), (8,7)

****Step 3: Choose a Cell and Determine Possible Numbers****

We select the cell at **** (6,1) **** for our next move.

- ****Numbers in Row 6**:** 2, 4, 6
- ****Numbers in Column 1**:** 1, 2, 3, 4, 6, 9
- ****Numbers in the 3×3 Block**:** 2, 4, 6, 7, 8
- ****Possible Numbers for (6,1)**:** ****5**** (since it's not present in the row, column, or block)

****Step 4: Make the Move****

Fill the number ****5**** at position **** (6,1) **** in the current state $S(i)$.

****Step 5: Ensure the Move Avoids Previously Failed Paths****

We avoid filling ****1**** at position (5,0) since it led to an unsolvable state. Our chosen move is different and moves us closer to a solution.

Next state: $[[2, 4, 7, 3, 6, 1, 8, 9, 5], [5, 6, 9, 7, 2, 8, 1, 4, 3], [3, 1, 8, 5, 9, 4, 7, 2, 6], [8, 2, 4, 6, 1, 5, 3, 7, 9], [7, 3, 6, 2, 8, 9, 4, 5, 1], [0, 0, 0, 0, 7, 0, 6, 8, 0], [0, 5, 2, 0, 0, 0, 0, 6, 4], [6, 0, 5, 1, 4, 0, 0, 0, 8], [0, 9, 0, 0, 5, 0, 2, 0, 7]]$

Table 24: An example of making an invalid move during state transition in Sudoku. The model fills 5 at position (6,1), which conflicts with 5 at position (7,2) as both 5s are in the same 3×3 subgrid.

A question from Sudoku 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 3×3 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:

$S(0) = [[0, 0, 6, 4, 0, 2, 0, 8, 0], [3, 7, 0, 0, 5, 0, 0, 9, 0], [2, 0, 8, 9, 0, 7, 4, 0, 0], [0, 0, 0, 0, 0, 9, 0, 0, 0], [0, 1, 0, 8, 0, 6, 0, 4, 3], [0, 0, 5, 7, 1, 0, 0, 0, 0], [0, 0, 0, 0, 0, 4, 0, 2, 7], [0, 8, 9, 2, 0, 0, 3, 0, 6], [0, 0, 2, 3, 0, 0, 8, 1, 0]]$

$L(0) = \text{Solvable}$

Two moves ago:

$S(i-2) = [[1, 9, 6, 4, 3, 2, 7, 8, 5], [3, 7, 4, 1, 5, 8, 6, 9, 2], [2, 5, 8, 9, 6, 7, 4, 3, 1], [0, 0, 0, 0, 0, 9, 0, 0, 0], [0, 1, 0, 8, 0, 6, 0, 4, 3], [0, 0, 5, 7, 1, 0, 0, 0, 0], [0, 0, 0, 0, 0, 4, 0, 2, 7], [0, 8, 9, 2, 0, 0, 3, 0, 6], [0, 0, 2, 3, 0, 0, 8, 1, 0]]$

$L(i-2) = \text{Solvable}$

One move ago:

$S(i-1) = [[1, 9, 6, 4, 3, 2, 7, 8, 5], [3, 7, 4, 1, 5, 8, 6, 9, 2], [2, 5, 8, 9, 6, 7, 4, 3, 1], [6, 0, 0, 0, 0, 9, 0, 0, 0], [0, 1, 0, 8, 0, 6, 0, 4, 3], [0, 0, 5, 7, 1, 0, 0, 0, 0], [0, 0, 0, 0, 0, 4, 0, 2, 7], [0, 8, 9, 2, 0, 0, 3, 0, 6], [0, 0, 2, 3, 0, 0, 8, 1, 0]]$

$L(i-1) = \text{Solvable}$

Current state:

$S(i) = [[1, 9, 6, 4, 3, 2, 7, 8, 5], [3, 7, 4, 1, 5, 8, 6, 9, 2], [2, 5, 8, 9, 6, 7, 4, 3, 1], [6, 2, 0, 0, 0, 9, 0, 0, 0], [0, 1, 0, 8, 0, 6, 0, 4, 3], [0, 0, 5, 7, 1, 0, 0, 0, 0], [0, 0, 0, 0, 0, 4, 0, 2, 7], [0, 8, 9, 2, 0, 0, 3, 0, 6], [0, 0, 2, 3, 0, 0, 8, 1, 0]]$

$L(i) = \text{Solvable}$

Explored next state:

$S(i+1) = [[1, 9, 6, 4, 3, 2, 7, 8, 5], [3, 7, 4, 1, 5, 8, 6, 9, 2], [2, 5, 8, 9, 6, 7, 4, 3, 1], [6, 2, 7, 0, 0, 9, 0, 0, 0], [0, 1, 0, 8, 0, 6, 0, 4, 3], [0, 0, 5, 7, 1, 0, 0, 0, 0], [0, 0, 0, 0, 0, 4, 0, 2, 7], [0, 8, 9, 2, 0, 0, 3, 0, 6], [0, 0, 2, 3, 0, 0, 8, 1, 0]]$

$L(i+1) = \text{Unsolvable}$

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.

Output from GPT-3.5 (Unsolvable Child)

Looking at the current state $S(i)$, we can see that the only empty cell in the third row and third column can be filled with the number 7 to form a valid solution. Therefore, the next move is to fill the cell at $S(i)[2][2]$ with the number 7.

Next state: $[[1, 9, 6, 4, 3, 2, 7, 8, 5], [3, 7, 4, 1, 5, 8, 6, 9, 2], [2, 5, 8, 9, 6, 7, 4, 3, 1], [6, 2, 7, 0, 0, 9, 0, 0, 0], [0, 1, 0, 8, 0, 6, 0, 4, 3], [0, 0, 5, 7, 1, 0, 0, 0, 0], [0, 0, 0, 0, 0, 4, 0, 2, 7], [0, 8, 9, 2, 0, 0, 3, 0, 6], [0, 0, 2, 3, 0, 0, 8, 1, 0]]$

Table 25: An example of transitioning to an unsolvable next state during state transition in Sudoku. The model simply fills 7, which is the same move as the unsolvable next state.

A question from Sudoku 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 3×3 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:

$S(0) = [[0, 0, 6, 4, 0, 2, 0, 8, 0], [3, 7, 0, 0, 5, 0, 0, 9, 0], [2, 0, 8, 9, 0, 7, 4, 0, 0], [0, 0, 0, 0, 0, 9, 0, 0, 0], [0, 1, 0, 8, 0, 6, 0, 4, 3], [0, 0, 5, 7, 1, 0, 0, 0, 0], [0, 0, 0, 0, 0, 4, 0, 2, 7], [0, 8, 9, 2, 0, 0, 3, 0, 6], [0, 0, 2, 3, 0, 0, 8, 1, 0]]$

$L(0) = \text{Solvable}$

Two moves ago:

$S(i-2) = [[1, 9, 6, 4, 3, 2, 7, 8, 5], [3, 7, 4, 1, 5, 8, 6, 9, 2], [2, 5, 8, 9, 6, 7, 4, 3, 1], [6, 2, 3, 5, 4, 9, 1, 7, 8], [9, 1, 7, 8, 2, 6, 5, 4, 3], [8, 4, 5, 7, 1, 0, 0, 0, 0], [0, 0, 0, 0, 0, 4, 0, 2, 7], [0, 8, 9, 2, 0, 0, 3, 0, 6], [0, 0, 2, 3, 0, 0, 8, 1, 0]]$

$L(i-2) = \text{Solvable}$

One move ago:

$S(i-1) = [[1, 9, 6, 4, 3, 2, 7, 8, 5], [3, 7, 4, 1, 5, 8, 6, 9, 2], [2, 5, 8, 9, 6, 7, 4, 3, 1], [6, 2, 3, 5, 4, 9, 1, 7, 8], [9, 1, 7, 8, 2, 6, 5, 4, 3], [8, 4, 5, 7, 1, 3, 0, 0, 0], [0, 0, 0, 0, 0, 4, 0, 2, 7], [0, 8, 9, 2, 0, 0, 3, 0, 6], [0, 0, 2, 3, 0, 0, 8, 1, 0]]$

$L(i-1) = \text{Solvable}$

Current state:

$S(i) = [[1, 9, 6, 4, 3, 2, 7, 8, 5], [3, 7, 4, 1, 5, 8, 6, 9, 2], [2, 5, 8, 9, 6, 7, 4, 3, 1], [6, 2, 3, 5, 4, 9, 1, 7, 8], [9, 1, 7, 8, 2, 6, 5, 4, 3], [8, 4, 5, 7, 1, 3, 9, 0, 0], [0, 0, 0, 0, 0, 4, 0, 2, 7], [0, 8, 9, 2, 0, 0, 3, 0, 6], [0, 0, 2, 3, 0, 0, 8, 1, 0]]$

$L(i) = \text{Unsolvable}$

Explored next state:

$S(i+1) = [[1, 9, 6, 4, 3, 2, 7, 8, 5], [3, 7, 4, 1, 5, 8, 6, 9, 2], [2, 5, 8, 9, 6, 7, 4, 3, 1], [6, 2, 3, 5, 4, 9, 1, 7, 8], [9, 1, 7, 8, 2, 6, 5, 4, 3], [8, 4, 5, 7, 1, 3, 9, 0, 0], [0, 0, 0, 0, 0, 4, 0, 2, 7], [0, 8, 9, 2, 0, 0, 3, 0, 6], [0, 0, 2, 3, 0, 0, 8, 1, 0]]$

$L(i+1) = \text{Unsolvable}$

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.

Output from GPT-3.5 (Backtracking Failure)

From the information provided, we can see that the current state is unsolvable, and the next explored state has also been proven to be unsolvable. To progress towards a valid solution, we need to avoid revisiting the explored failed path.

Looking at the current state, we see that the empty cell in row 5, column 9 (0-based index) needs to be filled to make progress. We can fill this cell with the number 6 to move towards a valid solution.

Therefore, the immediate next move is:

Fill: (5, 8) -> 6

Next state: $[[1, 9, 6, 4, 3, 2, 7, 8, 5], [3, 7, 4, 1, 5, 8, 6, 9, 2], [2, 5, 8, 9, 6, 7, 4, 3, 1], [6, 2, 3, 5, 4, 9, 1, 7, 8], [9, 1, 7, 8, 2, 6, 5, 4, 3], [8, 4, 5, 7, 1, 3, 9, 0, 6], [0, 0, 0, 0, 0, 4, 0, 2, 7], [0, 8, 9, 2, 0, 0, 3, 0, 6], [0, 0, 2, 3, 0, 0, 8, 1, 0]]$

Table 26: An example of failure to return to the parent state during state transition in Sudoku. The model continue to fill 6 from an unsolvable state instead of returning to the parent state $S(i-1)$.

A question from Sudoku 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 3×3 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:

$S(0) = [[0, 3, 5, 0, 0, 0, 8, 0, 1], [0, 0, 0, 0, 4, 0, 0, 0, 0], [2, 7, 0, 9, 0, 0, 6, 0, 3], [0, 8, 0, 3, 1, 0, 9, 0, 4], [0, 0, 9, 5, 2, 7, 0, 0, 0], [0, 1, 7, 0, 0, 9, 0, 6, 0], [0, 0, 8, 7, 0, 0, 2, 5, 0], [9, 0, 0, 0, 0, 5, 0, 1, 0], [0, 0, 4, 2, 0, 0, 0, 8, 6]]$

$L(0) = \text{Solvable}$

Two moves ago:

$S(i-2) = [[4, 3, 5, 6, 7, 2, 8, 9, 1], [8, 9, 6, 1, 4, 3, 7, 2, 5], [2, 7, 1, 9, 5, 8, 6, 4, 3], [5, 8, 2, 3, 1, 6, 9, 7, 4], [6, 4, 9, 5, 2, 7, 1, 3, 8], [3, 1, 7, 4, 8, 9, 5, 6, 2], [1, 0, 8, 7, 0, 0, 2, 5, 0], [9, 0, 0, 0, 0, 5, 0, 1, 0], [0, 0, 4, 2, 0, 0, 0, 8, 6]]$

$L(i-2) = \text{Solvable}$

One move ago:

$S(i-1) = [[4, 3, 5, 6, 7, 2, 8, 9, 1], [8, 9, 6, 1, 4, 3, 7, 2, 5], [2, 7, 1, 9, 5, 8, 6, 4, 3], [5, 8, 2, 3, 1, 6, 9, 7, 4], [6, 4, 9, 5, 2, 7, 1, 3, 8], [3, 1, 7, 4, 8, 9, 5, 6, 2], [1, 6, 8, 7, 0, 0, 2, 5, 0], [9, 0, 0, 0, 0, 5, 0, 1, 0], [0, 0, 4, 2, 0, 0, 0, 8, 6]]$

$L(i-1) = \text{Solvable}$

Current state:

$S(i) = [[4, 3, 5, 6, 7, 2, 8, 9, 1], [8, 9, 6, 1, 4, 3, 7, 2, 5], [2, 7, 1, 9, 5, 8, 6, 4, 3], [5, 8, 2, 3, 1, 6, 9, 7, 4], [6, 4, 9, 5, 2, 7, 1, 3, 8], [3, 1, 7, 4, 8, 9, 5, 6, 2], [1, 6, 8, 7, 9, 0, 2, 5, 0], [9, 0, 0, 0, 0, 5, 0, 1, 0], [0, 0, 4, 2, 0, 0, 0, 8, 6]]$

$L(i) = \text{Unsolvable}$

Explored next state:

$S(i+1) = [[4, 3, 5, 6, 7, 2, 8, 9, 1], [8, 9, 6, 1, 4, 3, 7, 2, 5], [2, 7, 1, 9, 5, 8, 6, 4, 3], [5, 8, 2, 3, 1, 6, 9, 7, 4], [6, 4, 9, 5, 2, 7, 1, 3, 8], [3, 1, 7, 4, 8, 9, 5, 6, 2], [1, 6, 8, 7, 9, 4, 2, 5, 0], [9, 0, 0, 0, 0, 5, 0, 1, 0], [0, 0, 4, 2, 0, 0, 0, 8, 6]]$

$L(i+1) = \text{Unsolvable}$

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.

Output from o1 (Sibling)

To determine the next move, let's analyze the progress made so far:

1. ****From $S(0)$ to $S(i-2)$ **** The first six rows of the puzzle were filled, making significant progress.
2. ****From $S(i-2)$ to $S(i-1)$ **** The number '6' was filled in row 6, column 1, advancing the puzzle.
3. ****From $S(i-1)$ to $S(i)$ **** The number '9' was filled in row 6, column 4. However, this led to an unsolvable state.
4. ****Explored $S(i+1)$ **** Filling '4' in row 6, column 5 also resulted in an unsolvable state.

****Conclusion**** Filling '9' at position [6][4] causes a conflict, making the puzzle unsolvable. To avoid this path and progress towards a solution, we should consider an alternative number for that cell.

****Analyzing possible numbers for position [6][4]****

- ****Numbers in the same row (row 6)**** 1, 6, 7, 8, 2, 5

- ****Numbers in the same column (column 4)**** 7, 4, 5, 1, 2, 8

- ****Numbers in the same 3×3 block**** 7, 2, 5

****Possible numbers for [6][4]**** Numbers from 1 to 9 not in the above lists: ****3**** and ****9****

Since '9' leads to an unsolvable state, we should try ****3**** at position [6][4].

****Next move**** Fill '3' in row 6, column 4.

Next state: $[[4, 3, 5, 6, 7, 2, 8, 9, 1], [8, 9, 6, 1, 4, 3, 7, 2, 5], [2, 7, 1, 9, 5, 8, 6, 4, 3], [5, 8, 2, 3, 1, 6, 9, 7, 4], [6, 4, 9, 5, 2, 7, 1, 3, 8], [3, 1, 7, 4, 8, 9, 5, 6, 2], [1, 6, 8, 7, 3, 0, 2, 5, 0], [9, 0, 0, 0, 0, 5, 0, 1, 0], [0, 0, 4, 2, 0, 0, 0, 8, 6]]$

Table 27: An example of making an additional move to a sibling state after backtracking during state transition in Sudoku. The model correctly backtracks by removing 9, but makes an additional move to fill 3 at position (6,4). This violates the instruction of making only one move each step.

Question	Model 1 (w/o Mixed Training)	Model 2 (w/ Mixed Training)
In Pascal's Triangle, the third row has four elements, and the sum of these elements is 8. What is the sum of the elements in the tenth row? (Note that we are counting the top row with a single 1 as the 0th row).	To determine the sum of the elements in the tenth row of Pascal's Triangle, we use the property that the sum of the elements in the n th row is 2^n . Since the top row is counted as the 0th row, the tenth row corresponds to the 9th row. Calculating 2^9, we get 512.	To determine the sum of the elements in the tenth row of Pascal's Triangle, we use the property that the sum of the elements in the n th row is 2^n . Given that the third row has a sum of 8, we can verify this by calculating $2^3 = 8$, which matches the given information. Therefore, the sum of the elements in the tenth row is $2^{10} = 1024$.
Determine the number of ways to arrange the letters of the word TEPEE.	Step 1: Identify the letters and their frequencies. The word TEPEE consists of the following letters: - T: 1 occurrence - E: 2 occurrences - P: 2 occurrences . Step 2: Apply the formula for permutations of a multiset. The formula for the number of distinct permutations of a word with repeated letters is: Number of permutations = $\frac{n!}{n_1! \times n_2! \times \dots \times n_k!}$ where: n is the total number of letters, n_1, n_2, \dots, n_k are the frequencies of the repeated letters. Step 3: Plug in the values. For the word TEPEE: $n = 5$ (since there are 5 letters). $n_1 = 1$ (for T). $n_2 = 2$ (for E). $n_3 = 2$ (for P). Number of permutations = $\frac{5!}{1! \times 2! \times 2!} = \frac{120}{1 \times 2 \times 2} = \frac{120}{4} = 30$.	1. Identify the Letters and Their Frequencies: The word TEPEE consists of 5 letters. The letters are: T, E, P, E, E. Frequencies: - T: 1 - E: 3 - P: 1 - C: 1 Note: The original problem didn't mention 'C', so we assume it's not part of the word. 2. Calculate the Total Number of Arrangements: The formula for permutations of a multiset is: Number of arrangements = $\frac{n!}{n_1! \times n_2! \times \dots \times n_k!}$ where n is the total number of items, and n_1, n_2, \dots, n_k are the frequencies of each distinct item. Applying this to TEPEE: Number of arrangements = $\frac{5!}{3! \times 1! \times 1!} = \frac{120}{6 \times 1 \times 1} = \frac{120}{6} = 20$

Table 28: Two examples of how the 1.5B model shows more evidence of verification and self-correction in its thought process after mixed training. Red denotes where the error starts to occur. Green denotes signs of verification or self-correction.