PHANTOM RECALL: When Familiar Puzzles Fool Smart Models

Anonymous ACL submission

Abstract

Large language models (LLMs) such as GPT, Gemini, and Claude often appear adept at solving classic logic puzzles—*but how much genuine reasoning underlies their answers?* Recent evidence suggests that these models frequently rely on memorized templates rather than reasoning from first principles. When puzzles are slightly modified, their performance collapses, revealing a striking fragility. In particular, we asked: *Have LLMs addressed these issues? To what extent? How about perturbations to other puzzles?Is there a general way of reformulating the prompt so that the models do better?* To examine these things systematically, we introduce **PHANTOM RECALL**, a benchmark comprising 25 well-known logic puzzles and 149 carefully designed perturbations that preserve reasoning structure but alter superficial details and solutions. We evaluate eleven leading LLMs and identify a recurring failure mode—phantom recall—where models confidently reproduce memorized solutions or spurious rationales that no longer fit the altered scenario. To probe and mitigate this issue, we contribute three tools: (i) an automated logical-equivalence judge to detect reasoning mismatches, (ii) a taxonomy of fine-grained reasoning error categories, and (iii) a prompting-based mitigation framework guided by these categories. Despite near-perfect accuracy on unmodified puzzles, models significantly underperform humans on perturbed ones, exhibiting both phantom recall and over-elaboration. Our findings reveal a crucial limitation: while LLMs excel at recalling reasoning, they often fail to re-reason when contextual cues shift—highlighting the gap between linguistic fluency and genuine logical understanding.

1 Introduction

It began with a puzzle–a riddle so simple that any schoolchild could solve it. Yet when posed to a large language model (LLM), the model answered with confidence–and was completely wrong. This amusing failure points to something deeper: despite their fluent prose and encyclopedic recall, LLMs often falter on tasks that require precise reasoning and faithful interpretation of instructions. Marcus (2024) catalogs numerous such failures, including the viral "strawberry problem" where models miscounted letters in a word—a task trivial for humans but revealing fundamental gaps in character-level reasoning. Statistician Colin Fraser has systematically documented this brittleness by "torturing" models with slight variations on familiar problems, demonstrating that minor perturbations often cause dramatic performance collapse as he shows in his twitter demonstrations on GPT Fails (Fraser, 2024).

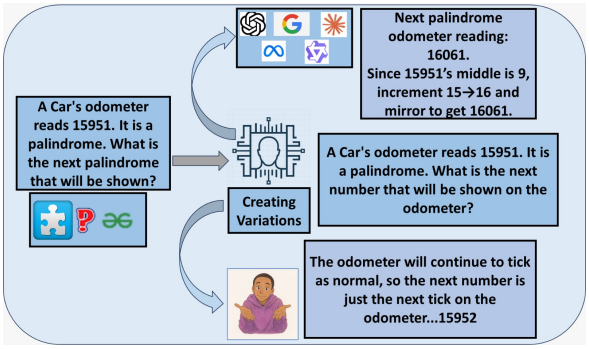


Figure 1: LLMs stumbled on this simple trivial puzzle variant—but a human can solve it instantly.

Over the past year, social media has amplified this phenomenon. Users frequently share screenshots of models stumbling on basic logic questions, misreading constraints, or hallucinating details never asked for. At first glance, these seem like trivial mistakes, but systematic studies confirm the pattern: even slight changes in wording, the addition of a constraint, or a twist on a familiar question can sharply degrade model accuracy (Srivastava et al., 2023; Lightman et al., 2023). These failures hint that models may be *pattern matchers rather than genuine reasoners*, leaning on statisti-

cal echoes of training data rather than recomputing solutions from first principles.

This issue is further complicated by benchmark contamination. Analyses reveal that benchmark familiarity can mask these weaknesses: models may reproduce memorized solutions to problems that appeared in training data, inflating apparent performance (Golchin and Surdeanu, 2023). Alignment effects such as *sycophancy* exacerbate the problem, with models optimizing for agreement and style over accuracy, often producing confident yet logically incoherent answers (Perez et al., 2022).

In this work, we turn this anecdotal observation into a rigorous evaluation paradigm. Inspired by Fraser's approach of systematic perturbation (Fraser, 2024), we compile a diverse set of riddles and logic puzzles, systematically perturb their constraints (basically changing the question to be as trivial as possible), and observe whether LLMs adjust their reasoning. We additionally construct "mirror image" variants designed to constrain the answer space, coaxing the model toward a correct solution with minimal wording changes. Our guiding question is simple: **when the puzzle changes, does the reasoning change too?** Our exploration of this main question yields a resounding answer: **More often than not, the model does not change its reasoning, leading to often humorous failures in logic on even the simplest of problems.**The overall problem scenario is illustrated in Figure 1. Our paper makes three major contributions:

1. We introduce a controlled suite of 25 classic puzzles with 149 carefully constructed variants that preserve the underlying reasoning template while changing the correct answer, revealing a recurring failure mode–*phantom recall*–where LLMs reproduce stock solutions that don't fit the instance. This isolates instance-level reasoning robustness under minimal perturbations.
2. Beyond final-answer accuracy, we provide (i) an automated conceptual-equivalence judge with answer verification auto-evaluator, (ii) a lightweight taxonomy of thinking-pattern configurations with fine-grained error labels (e.g., deductive failure, cascading error, improper elimination), and (iii) an LLM-aided step error classification autoevalutor that mirrors human step-analysis to surface where and why chains break—enabling model-agnostic, low-effort diagnosis of reasoning behavior.
3. Across 11 LLMs (6 closed models and 5 open-source models), SOTA systems approach perfect accuracy on base puzzles but drop sharply on perturbed variants, with errors dominated by phantom recall and over-elaboration; "thinking/structured" prompting reliably improves outcomes yet does not close the robustness gap–highlighting concrete targets for future training-time interventions.

2 Related Work

2.1 Puzzle and its variation solving SOTA

Puzzle-solving offers a window into LLMs' logical abilities. Giadikiaroglou et al. (2024) divide puzzles into (1) rule-based and (2) rule-less categories. Rule-less types include riddles (Lin et al., 2021), MCQs (Zhao and Anderson, 2023), programming puzzles (Schuster et al., 2021), and commonsense reasoning puzzles (Gu et al., 2023); our focus is on rule-based puzzles. Prior efforts on rule-based puzzles span Sudoku (Noever and Burdick, 2021), Rubik's Cube, the 8-puzzle, Game of 24 (Yao et al., 2023), crosswords (Yao et al., 2023), chess puzzles (Feng et al., 2023), card games (Gupta, 2023), BoardgameQA (Kazemi et al., 2023), and Lateral Thinking Puzzles (Huang et al., 2023). Similarly, Mitra and Baral (2015) proposed a grid-based puzzle dataset, and (Dziri et al., 2023) studied compositionality in LLMs using Grid Puzzle. Besides that, some minimal works have been done on compositional variation on puzzle. Williams and Huckle (2024) showed that LLMs can easily get wrong on the puzzle variation. But reasoning analysis of why LLMs can easily get wrong, has not been done yet. Getting motivated by that, we want to explore the detailed reasoning chain, where LLMs are getting wrong with a proposed puzzle variation dataset.

2.2 Step-by-Step analysis of reasoning chains generated by Language Models SOTA

Reference-free evaluation, which is independent of gold reasoning chains, has been the focus of earlier studies (Dalvi et al., 2021; Saparov and He, 2022; Han et al., 2022). ROSCOE (Golovneva et al., 2023) provides a broad metric suite to evaluate reasoning quality across correctness, informativeness, consistency, and coherence. ReCEval (Prasad et al., 2023) narrows the focus to incorrect-answer detection by analyzing the correctness and informativeness of intermediate steps. LLM Reasoners (Hao et al., 2024) offers a framework for fine-grained,

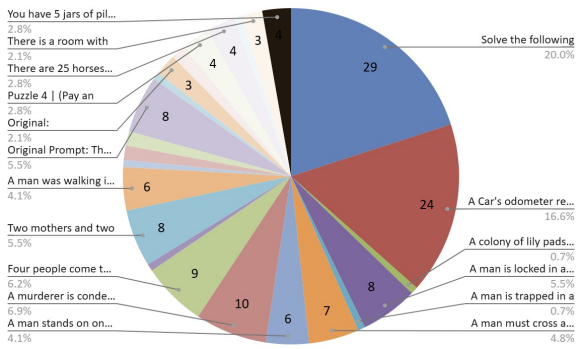


Figure 2: Distribution of standard puzzle and variations.

step-by-step assessment of large language models' reasoning. Ling et al. (2023) introduces Natural Program, a natural-language deductive format that breaks verification into sequential substeps. Tyen et al. (2023) examines LLM limitations in spotting reasoning errors while showing strong performance at fixing them when error locations are provided. Li et al. (2022) presents DIVERSE, a three-stage method to identify and correct errors throughout a reasoning chain. Although prior work proposes valuable evaluation frameworks (Patel et al., 2024; Parmar et al., 2024; Tyagi et al., 2024; Mishra et al., 2025), they are not tailored to puzzle variation. In contrast, our approach employs simple yet effective soundness and correctness metrics to yield detailed insights into step-wise standard puzzle variation reasoning errors and to enable scalable, low-effort evaluation via an LLM-based framework.

3 Dataset

3.1 Dataset Construction

We began with a base set of 25 classic riddles and logic problems, sourced from a combination of Wikipedia entries on famous puzzles, online puzzle forums, and brainteaser collections. To diversify the set, we also generated riddles with ChatGPT based on human-authored examples, then manually reviewed and edited them before inclusion. For each base puzzle, we manually created **perturbed variants** by altering constraints, numerical values, or other problem details. The key principle was to preserve the *reasoning structure* while forcing a different final answer, ensuring that models could not rely on memorization. These perturbations allow us to test whether models genuinely recompute solutions rather than simply recalling previously seen ones. Pie chart of Figure 2 shows the number of standard puzzle and variations.

3.2 Mirror-Image Dataset

To further probe reasoning, we developed a **mirror-image subset** of the dataset. Here, 43 of the more open-ended puzzle variations were rewritten to constrain the possible answer space. For example, an open-ended riddle asking "What could the object be?" might be reformulated into a multiple-choice format with three plausible candidates. This allows us to measure whether additional structure or hints can recover correct reasoning when the free-form version fails. It is described in App. A in a great details with several examples.

3.3 Validation and Quality Control

Every riddle and its variants underwent a multi-step validation process:

1. **Instruction-Following Check:** We verified that models interpreted the question correctly rather than answering an unintended reformulation.
2. **Reasoning Efficiency:** We flagged cases where the model's solution path was overly convoluted, suggesting inefficiency or spurious reasoning.
3. **Ambiguity Resolution:** If a model produced an unintended but logically valid answer, we revised the puzzle wording to eliminate multiple interpretations.
4. **Consistency Assessment:** We documented cases where models triggered self-verification (e.g., "Let me double-check"), as these often revealed reasoning uncertainty.

3.4 Answer Verification Auto-Evaluator

We developed an LLM-based auto-evaluator to verify the final answer for each data point. Each predicted answer is compared with the gold answer. If the answer matches, the instance is assigned a label of *TRUE*, and *FALSE* otherwise. The human verification result is described in App. B. Answer verification auto-evaluator system prompt is described in App. C. The accuracy comparison on closed source model is shown in Figure 3. It states that our LLM based Answer verification auto-evaluator gives ∼96% similarity with human annotations.

4 Evaluation of Reasoning Chains

4.1 Manual Evaluation of Reasoning Chains

To move beyond automated metrics and gain a deeper understanding of the failure modes in modern language model reasoning, we performed a

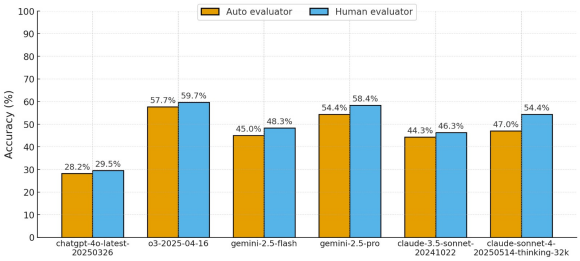


Figure 3: Grouped bar chart: Auto vs Human accuracy on closed source models

rigorous qualitative analysis of their generated outputs. Our analysis corpus consisted of 50 reasoning chains generated for variation puzzles. These were sampled from five distinct open-source models: (LLaMa(Touvron et al., 2023), InternLM(Cai et al., 2024), Qwen(Bai et al., 2023), Phi(Abdin et al., 2024), and Mistral(Jiang et al., 2023)), with 10 chains evaluated per model to ensure a comparative perspective. The lengths of the chains generated by the different Open-Source LLMs analyzed are discussed in the table in App. D. Our manual evaluation protocol involved two stages:

1. **Logical Step Segmentation:** First, each complete reasoning chain was deconstructed into its constituent sentences. Each sentence was treated as an individual reasoning step, allowing for a granular assessment of the model's logical progression.
2. **Premise-Conclusion Analysis:** Second, each segmented step was meticulously examined to isolate its core logical components: the premise(s) upon which the reasoning was based, and the resulting conclusion. We then assessed the validity of this inference, determining whether the conclusion logically followed from the premise and whether the premise itself was factually correct given the puzzle's constraints and the preceding steps.

The primary objective of this methodology was to create a comparative profile of reasoning errors, enabling us to identify systemic weaknesses and differentiate the qualitative performance of each model. Additionally, to ensure that the error categorization was performed with the utmost attention to detail, the human annotators were given the annotation guidelines in App. E to adhere to.

4.2 Proposed Reasoning Configuration Error Categories

Based on our analysis, we developed a framework to systematically classify the errors within each reasoning step. This framework consists of five high-level categories based on the soundness of the premise and conclusion:

- **Logically Sound Step:** A step where a correct conclusion is validly derived from a correct premise.
- **Deductive Failure:** A step where a correct premise leads to an incorrect conclusion due to flawed reasoning.
- **Compounded Failure:** A step where an incorrect premise leads to an incorrect conclusion.
- **Fortuitous Correctness:** An interesting case where a flawed premise coincidentally results in a correct conclusion.
- **Declarative Statement:** A sentence that restates a clue or a previous finding without drawing a new conclusion.

To pinpoint the root causes of failures, we further analyzed the origins of flawed premises and the nature of incorrect conclusions. **Sources of Flawed Premises:** We identified two sources for incorrect premises: (1) **Evidence-based**, where information is drawn directly from the puzzle's clues, and (2) **Internally-derived**, where the premise is inferred from previous steps. Errors associated with these sources include:

- **Evidence Misrepresentation:** The premise fabricates or distorts factual information from the clues.
- **Insufficient Evidence:** The premise is factually correct but inadequate to support the drawn conclusion.
- **Unsupported Assumption:** The model introduces a speculative premise not directly given by or derived from the clues.
- **Cascading Error:** An error from a previous conclusion is carried forward as a flawed premise.
- **Faulty Postulate:** An explicitly stated assumption is derived incorrectly from prior information.

Types of Flawed Conclusions: We classified incorrect conclusions into three primary types:

- **Consequential Error:** The conclusion is incorrect as a direct result of a flawed premise.

- **Improper Candidate Elimination:** The model fails to correctly discard all invalid options, a common failure in constraint-based puzzles.
- **General Logical Fallacy:** A catch-all for other invalid deductions where the reasoning itself is erroneous, independent of the premise's correctness.

For clarity, we summarize this error taxonomy in Table 1.

4.3 Step Error Classification auto-evaluator

While our manual analysis provides a deep, qualitative understanding of reasoning failures, its meticulous nature makes it challenging to scale across our entire dataset. To analyze the distribution of errors at a larger scale–a crucial step for understanding the systemic shortcomings of LLMs in creative reasoning–we developed an LLM-based automated evaluator.

We termed this system the **"Step Error Classification auto-evaluator"**. It utilizes the GPT-4o model to automate the error identification and categorization process for riddle-solving reasoning chains. The auto-evaluator is guided by a comprehensive prompt containing:

1. **System Instructions:** A detailed set of rules and sequential steps, mirroring our manual evaluation protocol, that instruct GPT-4o on how to deconstruct and analyze a reasoning chain. This also specifies the required JSON output format for parsing.
2. **Domain Knowledge:** A complete definition of our error classification framework, including the five high-level categories and their associated sub-categories. To minimize ambiguity, we provide a clear preference order for classifying errors that might fit multiple definitions.
3. **A Guiding Exemplar:** A one-shot example consisting of a riddle, its ground truth answer, a model-generated reasoning chain, and our corresponding manual evaluation, which serves as a template for the expected output.

The full prompt structure for the Auto-evaluator is provided in App F.

Using this Auto-evaluator, we analyzed a total of 50 reasoning chains generated by our five open-source models for 10 unique riddles. To validate the fidelity of this automated approach, we compared its output against our human-annotated ground truth. We randomly sampled 50 chains from our manually evaluated set and found that the Auto-evaluator achieved an average 71% coincidence with human evaluators. The overall architecture of our paper is shown in Fig 4.

5 Experimental Setup

5.1 Models

We evaluate a range of open-source LLMs, including **LLaMa(Touvron et al., 2023), InternLM(Cai et al., 2024), Qwen(Bai et al., 2023), Phi(Abdin et al., 2024), and Mistral(Jiang et al., 2023)**, on a proprietary dataset of riddles and logic problems. Our dataset consists of 25 base puzzles and a corresponding set of **perturbed variants**. Each variant is manually created by altering constraints or numerical values to change the gold answer, , while preserving the underlying reasoning structure. This design forces models to recompute solutions rather than rely on memorizing answers to known problems.

To evaluate each model, we use the **Zero-shot Chain-of-Thought** setting (Kojima et al., 2023). The model receives a natural language instruction and a puzzle variant as input and is prompted to generate a reasoning chain before predicting a final answer. The evaluation was conducted on the OpenAI, Google, and Anthropic model versions released between October 2024 and May 2025, with a temperature setting of 0 for deterministic predictions. Inference for the open-source models was run locally using 2 A100 GPU units.

5.2 Metrics

We use **accuracy** to demonstrate the capability of LLMs in solving riddles based on their ability to predict the correct answer. To calculate this metric, we compare the LLM-generated final answers with the established ground truth solution for each riddle.

The predicted answers and the ground truth solutions are typically in the form of a single word or a short phrase. We perform a normalized **Exact Match (EM)** to compare the two. Before comparison, both the predicted and the ground truth answers are standardized by converting them to lowercase and removing any leading/trailing whitespace or punctuation. An LLM's response is marked as correct only when the normalized text matches the normalized solution exactly. This process ensures

Category	Source	Sub-Category	Description
Deductive Failure, Fortuitous Correctness, Compounded Failure, Declaration	From premise (e.g., 'Considering first two points...')	(1) Evidence Misrepresentation	Premise fabricates or distorts factual information.
		(2) Insufficient Evidence	Premise correct but inadequate to support conclusion.
		(3) Unsupported Assumption	Model introduces premise not given by clues.
	Derived Conclusions using premises not given.	(4) Cascading Error	Error from previous conclusion carried forward.
		(5) Faulty Postulate	Assumption derived incorrectly from prior info.
Compounded Failure	Derived using premise (taken directly or derived)	(a) General Logical Fallacy	Invalid deductions with erroneous reasoning.
		(b) Consequential Error	Conclusion incorrect due to flawed premise.
		(c) Improper Candidate Elimination	Fails to discard invalid options.

Table 1: **Error taxonomy for reasoning failures.**

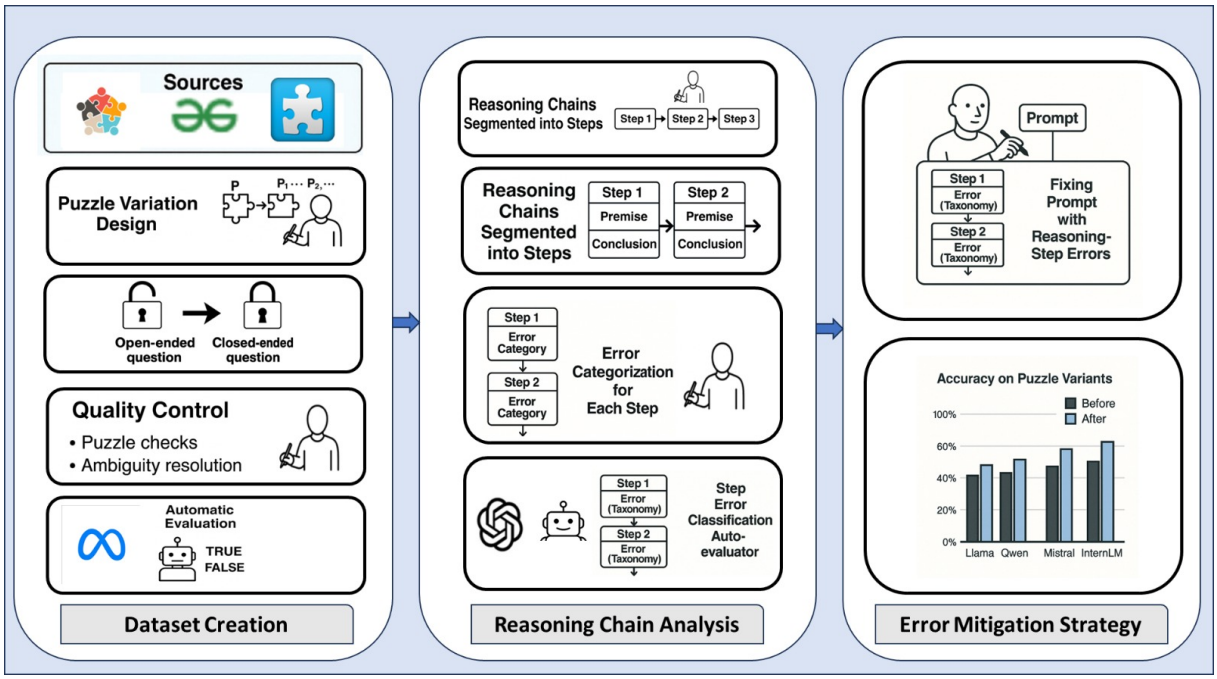


Figure 4: Overview of our three major contributions and their interconnections. The benchmark enables systematic reasoning chain analysis, which informs the development of error mitigation strategies. These strategies are validated on the benchmark, creating a comprehensive framework.

that minor formatting differences do not impact the evaluation of the model's core reasoning ability.

6 Results and Analysis

6.1 Objective Evaluation

For the objective evaluation, our primary metric was the accuracy of the final answer provided by each model with the help of Answer verification auto-evaluator. This quantitative assessment serves as a baseline for performance, measuring a model's ability to arrive at the correct solution, irrespective of the reasoning path taken. The results, summarized in Figure 5, reveal a distinct performance hierarchy among the evaluated models. Qwen emerged as the top performer, achieving **74** correct final answers. It was followed by Phi and LLaMA, which secured **48** and **46** correct answers, respectively. Mistral and InternLM recorded the lowest scores in this evaluation, with Mistral providing **33** correct answers while InternLM provided **40**. These empirical results offer a clear, quantitative comparison of the models' problem-solving capabilities on our dataset.

Figure 6 compares the distribution of step-wise reasoning outcomes for five open-source LLMs–LLaMA, Qwen, InternLM, Mistral, and Phi. Each stacked bar shows the proportion of steps labeled as *Logically Sound Step*, *Deductive Failure*, *Compounded Failure*, *Fortuitous Correctness*, and

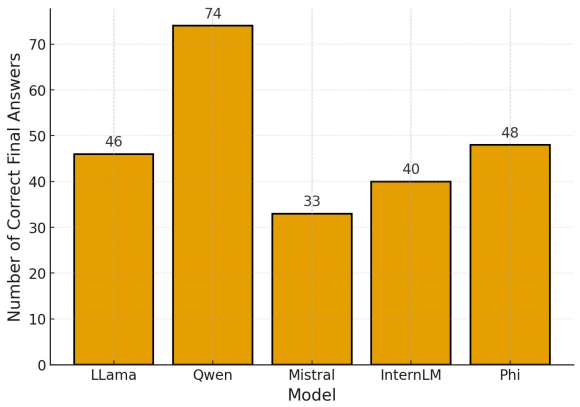


Figure 5: Performance of five different open source LLMs (LLama 3.1, Phi 4, Mistral 7B, Qwen 2.5 7B, and InternLM) in terms of accuracy on the Phantom Recall dataset.

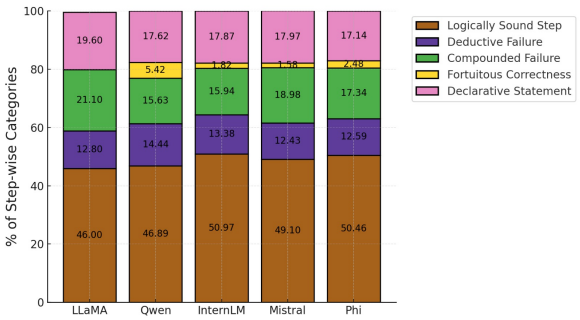


Figure 6: Comparative distribution of step-wise reasoning errors in 5 open source LLMs

Declarative Statement. Across models, logically sound steps dominate (∼46–51%), indicating a substantial fraction of coherent intermediate reasoning. Deductive and compounded failures together comprise roughly 25–35%, revealing persistent weaknesses that can derail full-chain correctness. Fortuitous correctness remains rare (<6%), while declarative (non-reasoning) statements are relatively stable at ∼17–19%. Overall, the figure highlights that although most steps are logically valid, deduction gaps and error propagation still limit end-to-end reasoning reliability.

Figure 7 and Figure 8 jointly reveal a consistent error pattern across LLaMA, Qwen, InternLM, Mistral, and Phi. At the *premise* level (Fig. 7), *Insufficient Evidence* dominates (24–27%), followed by *Unsupported Assumption* (21–23%) and *Evidence Misrepresentation* (18–22%); *Cascading Error* contributes 17–20%, while *Faulty Postulate* is least common (13–15%). At the *conclusion* level (Fig. 8), *Consequential Error* is most prevalent (5–38%), with *Improper Candidate Elimination* close behind (33–38%) and *General Logical Fallacy* comprising the remainder (27–29%). Together, the figures indicate that failures often originate from evidence insufficiency and assumption-based reasoning early in the chain, which then propagate to the final decision; among the models, InternLM (and to a lesser extent Qwen) shows slightly more balanced/robust distributions, underscoring the need for better evidence aggregation and end-of-chain verification.

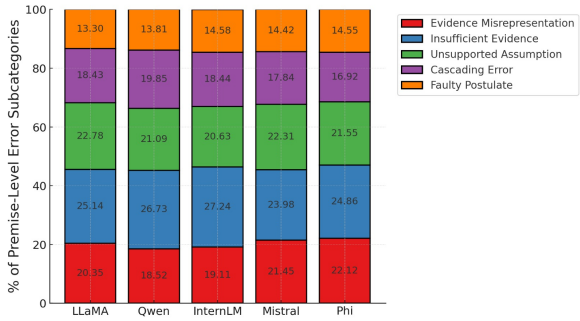


Figure 7: Comparative distribution of premise level errors in 5 open source LLMs

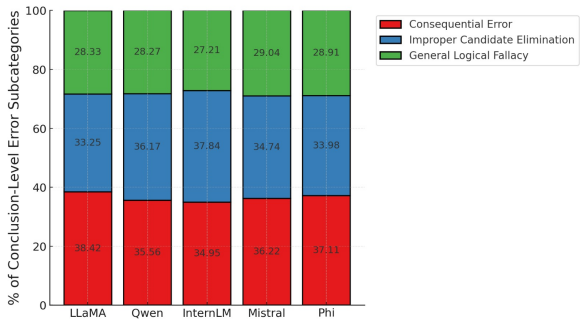


Figure 8: Comparative distribution of conclusion level errors in 5 open source LLMs

6.2 Thinking Chain Evaluation

To systematically evaluate the logical integrity of each thinking chain, we employed a granular, step-by-step analysis. Each chain was first decomposed into its constituent steps. Following a structured annotation guideline, we designated each step as having a **'premise'** and a **'conclusion'**. This dichotomy enabled a two-level error analysis. First, the premises were scrutinized for any logical fallacies, factual inaccuracies, or reasoning flaws, which were classified as **'premise-level errors'**. Subsequently, the conclusion was evaluated, not only for its correctness but also for its logical dependency on the preceding premises. Errors identified at this final stage were classified as **'conclusion-**

level errors'. This methodical approach allowed for the precise localization of failure points within the model's reasoning process.

6.3 Discussion on Error-Mitigation Strategies

Our primary strategy uses **prompts with explicit prohibitory instructions** to establish clear negative constraints that prune the response space. We tailored these to target specific error categories: (1) **Hallucination:** "Do not use any external knowledge" ensures faithfulness to context; (2) **Speculation:** "Do not make assumptions about missing information" prevents unwarranted inferences; (3) **Formatting:** "Do not output any commentary" and "Your response must be only the valid JSON object" enforce parsability. This shifts the model from generative to controlled behavior.

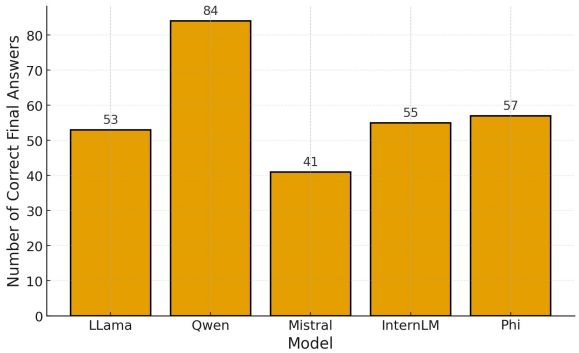


Figure 9: Performance of five different open source LLMs in terms of accuracy on the Phantom Recall dataset with modified prompts.

Our error-mitigation strategy consistently improved accuracy across all models on the 149-puzzle dataset. Accuracy gains ranged from 4.7 percentage points (LLaMA: 30.9%→35.6%) to 10.1 points (InternLM: 26.8%→36.9%), with Qwen (49.7%→56.4%), Mistral (22.1%→27.5%), and Phi (32.2%→38.3%) showing intermediate improvements of 5.4–6.7 points.

7 Conclusion

We set out to investigate whether large language models (LLMs) have truly addressed their reasoning fragility—beyond memorized templates and benchmark familiarity. Our findings across eleven leading models suggest that while newer LLMs demonstrate marginal improvements in reasoning coherence, **they have not fundamentally solved the problem**. First, regarding *whether LLMs have addressed these issues*, our results show that the failure mode we term **phantom recall** persists across all systems, indicating that reasoning robustness remains largely unaddressed. Second, on *the extent of improvement*, performance gains are narrow and model-specific, often limited to the original puzzle formulations. Third, in examining *whether models made specific fixes*, we find evidence of overfitting: some models reproduce polished but instance-bound reasoning patterns rather than general strategies. Fourth, concerning *generalization*, when confronted with minimally perturbed puzzles that preserve logical structure, models rarely adjust their reasoning—demonstrating that their "fixes" lack true transferability. Finally, for *perturbations to other puzzles*, the degradation in accuracy and logical soundness remains substantial even under small semantic or structural shifts. Overall, our study reveals that while LLMs can **remember reasoning**, they often fail to **re-reason**. Their responses exhibit linguistic fluency without genuine logical flexibility. We hope that **PHANTOM RECALL** serves as both a diagnostic tool and a call to action for designing future models and training strategies that move beyond pattern matching toward context-grounded, adaptive reasoning.

Limitations

While our benchmark provides valuable insights into LLM reasoning capabilities through logic puzzles, several constraints should be acknowledged. First, our dataset comprises 25 base puzzles with 149 systematically generated variants. Although this scale enables controlled analysis of model behavior across puzzle variations, a larger and more diverse puzzle set would strengthen claims about generalization. The current scope, however, allows for thorough manual verification and detailed error analysis that would be prohibitively expensive at larger scales.

Second, we evaluate 8 state-of-the-art models spanning both open-source and proprietary systems. While these models represent the current frontier of language model capabilities and include the most widely deployed systems, the rapid pace of model development means that newer architectures may exhibit different reasoning patterns. Our selection prioritizes models with substantial adoption in research and practice, ensuring our findings remain relevant to the broader community.

Third, our focus on logic puzzles, while offering clear ground truth and interpretability, represents

only one facet of reasoning ability. Performance on these structured tasks may not fully capture model behavior on open-ended reasoning or real-world problem-solving scenarios. However, the controlled nature of logic puzzles enables precise identification of failure modes and reasoning patterns that are difficult to isolate in less structured domains.

Finally, we acknowledge that our evaluation is conducted at a single point in time. As models continue to improve and prompting strategies evolve, the absolute performance numbers reported here may be superseded. Nevertheless, the relative comparisons between models and the qualitative insights into reasoning failures provide enduring value for understanding current limitations and informing future development.

Ethics Statement

We have used ChatGPT and Grammarly, two AI assistants, to reword statements and fix grammar mistakes.

References

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Hassan Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Singh Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Caio C'esar Teodoro Mendes, Weizhu Chen, Vishrav Chaudhary, Parul Chopra, and 69 others. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *ArXiv*, abs/2404.14219.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenhang Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, K. Lu, and 31 others. 2023. Qwen technical report. *ArXiv*, abs/2309.16609.

Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, Xiao wen Dong, Haodong Duan, Qi Fan, Zhaoye Fei, Yang Gao, Jiaye Ge, Chenya Gu, Yuzhe Gu, Tao Gui, and 81 others. 2024. Internlm2 technical report. *ArXiv*, abs/2403.17297.

Bhavana Dalvi, Peter Alexander Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. 2021. Explaining answers with entailment trees. In *Conference on Empirical Methods in Natural Language Processing*.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jian, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Sean Welleck, Xiang Ren, Allyson Ettinger, Zaïd Harchaoui, and Yejin Choi. 2023. Faith and fate: Limits of transformers on compositionality. *ArXiv*, abs/2305.18654.

Xidong Feng, Yicheng Luo, Ziyan Wang, Hongrui Tang, Mengyue Yang, Kun Shao, David Henry Mguni, Yali Du, and Jun Wang. 2023. Chessgpt: Bridging policy learning and language modeling. *ArXiv*, abs/2306.09200.

Colin Fraser. 2024. Gpt fails. Personal blog/Twitter demonstrations. Collection of systematic LLM failure cases through problem variants.

Panagiotis Giadikiaroglou, Maria Lymperaiou, Giorgos Filandrianos, and G. Stamou. 2024. Puzzle solving using reasoning of large language models: A survey. *ArXiv*, abs/2402.11291.

Shahin Golchin and Mihai Surdeanu. 2023. Benchmarks are not perfect: The effects of data contamination on model evaluation. *arXiv preprint arXiv:2304.10460*.

O. Yu. Golovneva, Moya Chen, Spencer Poff, Martin Corredor, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2023. Roscoe: A suite of metrics for scoring step-by-step reasoning. In *International Conference on Learning Representations*.

Zhouhong Gu, Zihan Li, Lin Zhang, Zhuozhi Xiong, Sihang Jiang, Xiaoxuan Zhu, Shusen Wang, Zili Wang, Jianchen Wang, Haoning Ye, Wenhao Huang, Yikai Zhang, Hongwei Feng, and Yanghua Xiao. 2023. Beyond the obvious: Evaluating the reasoning ability in real-life scenarios of language models on life scapes reasoning benchmark (lsr-benchmark). *ArXiv*, abs/2307.05113.

Akshat Gupta. 2023. Are chatgpt and gpt-4 good poker players? - a pre-flop analysis. *ArXiv*, abs/2308.12466.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Luke Benson, Lucy Sun, Ekaterina Zubova, Yujie Qiao, Matthew Burtell, David Peng, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, and 7 others. 2022. Folio: Natural language reasoning with first-order logic. *ArXiv*, abs/2209.00840.

Shibo Hao, Yi Gu, Haotian Luo, Tianyang Liu, Xiyan Shao, Xinyuan Wang, Shuhua Xie, Haodi Ma, Adithya Samavedhi, Qiyue Gao, Zhen Wang, and Zhiting Hu. 2024. Llm reasoners: New evaluation, library, and analysis of step-by-step reasoning with large language models. *ArXiv*, abs/2404.05221.

Shulin Huang, Shirong Ma, Yinghui Li, Mengzuo Huang, Wuhe Zou, Weidong Zhang, and Haitao Zheng. 2023. Lateval: An interactive llms evaluation benchmark with incomplete information from lateral thinking puzzles. In *International Conference on Language Resources and Evaluation*.

Albert Qiaochu Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *ArXiv*, abs/2310.06825.

Mehran Kazemi, Quan Yuan, Deepti Bhatia, Najoung Kim, Xin Xu, Vaiva Imbrasaite, and Deepak Ramachandran. 2023. Boardgameqa: A dataset for natural language reasoning with contradictory information. *ArXiv*, abs/2306.07934.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large language models are zero-shot reasoners. *Preprint*, arXiv:2205.11916.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, B. Chen, Jian-Guang Lou, and Weizhu Chen. 2022. Making language models better reasoners with step-aware verifier. In *Annual Meeting of the Association for Computational Linguistics*.

Spencer Lightman and 1 others. 2023. Let's think step by step: Evaluating reasoning in large language models. *arXiv preprint arXiv:2305.20050*.

Bill Yuchen Lin, Ziyi Wu, Yichi Yang, Dong-Ho Lee, and Xiang Ren. 2021. Riddlesense: Reasoning about riddle questions featuring linguistic creativity and commonsense knowledge. In *Findings*.

Z. Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. 2023. Deductive verification of chain-of-thought reasoning. *ArXiv*, abs/2306.03872.

Gary Marcus. 2024. Alphaproof, alphageometry, chatgpt o1, and the strawberry problem. Gary Marcus Substack. Accessed: 2025-10-05.

Venkatesh Mishra, Bimsara Pathiraja, Mihir Parmar, Sat Chidananda, Jayanth Srinivasa, Gaowen Liu, Ali Payani, and Chitta Baral. 2025. Investigating the shortcomings of llms in step-by-step legal reasoning. In *North American Chapter of the Association for Computational Linguistics*.

Arindam Mitra and Chitta Baral. 2015. Learning to automatically solve logic grid puzzles. In *Conference on Empirical Methods in Natural Language Processing*.

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

Mihir Parmar, Nisarg Patel, Neeraj Varshney, Mutsumi Nakamura, Man Luo, Santosh Mashetty, Arindam Mitra, and Chitta Baral. 2024. Logicbench: Towards systematic evaluation of logical reasoning ability of large language models. In *Annual Meeting of the Association for Computational Linguistics*.

Nisarg Patel, Mohith Kulkarni, Mihir Parmar, Aashna Budhiraja, Mutsumi Nakamura, Neeraj Varshney, and Chitta Baral. 2024. Multi-logieval: Towards evaluating multi-step logical reasoning ability of large language models. *ArXiv*, abs/2406.17169.

Ethan Perez and 1 others. 2022. Discovering language model behaviors with model-written evaluations. *arXiv preprint arXiv:2206.12388*.

Archiki Prasad, Swarnadeep Saha, Xiang Zhou, and Mohit Bansal. 2023. Receval: Evaluating reasoning chains via correctness and informativeness. In *Conference on Empirical Methods in Natural Language Processing*.

Abulhair Saparov and He He. 2022. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *ArXiv*, abs/2210.01240.

Tal Schuster, A. Kalyan, Oleksandr Polozov, and Adam Tauman Kalai. 2021. Programming puzzles. *ArXiv*, abs/2106.05784.

Aarohi Srivastava and 1 others. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2304.15004*.

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

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 *Conference on Empirical Methods in Natural Language Processing*.

Gladys Tyen, Hassan Mansoor, Peter Chen, Tony Mak, and Victor Carbune. 2023. Llms cannot find reasoning errors, but can correct them! In *Annual Meeting of the Association for Computational Linguistics*.

Sean Williams and James Huckle. 2024. Easy problems that llms get wrong. *ArXiv*, abs/2405.19616.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *ArXiv*, abs/2305.10601.

Jin Zhao and Carolyn Jane Anderson. 2023. Solving and generating npr sunday puzzles with large language models. In *International Conference on Innovative Computing and Cloud Computing*.

A Open Ended Question to Closed Ended Question

The process of transforming riddles from open-ended to closed-ended formats was conducted through systematic iterative refinement by the two principal annotators. This transformation represents a critical methodological challenge, as it requires maintaining the essential cognitive and reasoning demands of the original riddle while fundamentally restructuring the response format to eliminate interpretive flexibility. The workflow proceeded through several carefully designed stages to ensure both consistency and validity in the transformation process.

Initially, one annotator proposed a modification to the final question posed in the riddle, reformulating it to constrain the response space while preserving the core reasoning challenge. This reformulation process required careful consideration of multiple factors: the question needed to eliminate the possibility of multiple valid interpretations, guide respondents toward a specific answer structure (such as binary choices, likelihood comparisons, or entity identification), and maintain the logical dependencies and inferential steps required in the original riddle. The annotator would explore various reformulation strategies, including converting "how" or "why" questions into "what is more likely" comparisons, transforming explanation requests into identification tasks, or restructuring open queries into forced-choice formats that preserved the underlying reasoning path.

Each modified riddle was then evaluated empirically by prompting the target LLMs and analyzing their generated responses. This evaluation phase was essential to the refinement process, as it revealed how models interpreted and responded to the reformulated questions in practice. The annotators examined these outputs systematically to identify any residual ambiguity in interpretation or reasoning paths that could lead to multiple valid answers. This examination included analyzing whether the models provided consistent answers across multiple trials, whether they exhibited uncertainty or hedging in their responses, and whether they demonstrated understanding of the intended logical structure of the riddle. Particular attention was paid to cases where models might exploit loopholes in the question wording, apply unexpected but technically valid interpretations, or demonstrate confusion about the intended answer format.

A riddle variant was classified as 'sufficiently closed-ended' when it satisfied one of two carefully defined criteria. First, the LLM responses demonstrated no ambiguity in interpretation, converging on a single clearly-defined answer with consistent reasoning across multiple evaluation trials. This criterion required not only that models select the correct answer, but that they do so for the right reasons, demonstrating that the question structure successfully constrained their reasoning process. Second, any remaining ambiguity could be attributed to factors independent of the question's wording itself—such as inherent semantic ambiguity in the riddle's premise, world knowledge gaps, or fundamental uncertainties in the scenario description—rather than to structural openness in the question format. This second criterion was important for distinguishing between ambiguity introduced by the transformation process and ambiguity that existed in the original riddle material itself.

Throughout this iterative testing-and-refinement cycle, the annotators maintained detailed records of proposed modifications, model responses, and the rationale for accepting or rejecting particular reformulations. When a proposed modification failed to adequately constrain the response space, the annotators engaged in collaborative discussion to diagnose the source of remaining ambiguity and propose alternative reformulations. This process continued until the annotators reached consensus that the closed-ended variant adequately constrained the solution space without fundamentally altering the original riddle's logical structure or cognitive demands. The consensus requirement ensured that transformations met high standards of quality and faithfulness to the source material, preventing idiosyncratic interpretations from influencing the final dataset.

The transformation process also involved careful consideration of how to preserve specific reasoning types across format changes. For instance, riddles requiring lateral thinking needed to maintain their misdirection elements even when converted to multiple-choice formats, while riddles based on wordplay or semantic ambiguity needed reformulations that preserved the relevant linguistic features. A snapshot of the process is shown for two examples in Figure 10, illustrating both the structural changes in question format and the preservation of core riddle content across the transformation.

Original Version

It's a hot rainy day and a brown hair man is standing on one side of a perennial river, with his dog on the other. There is only one option to cross the river— a bridge. It is heavily raining outside. The man calls his dog, who immediately crosses the river without getting a single fur wet and without using a bridge. On the other hand the man is completely wet. How did the dog do it?

Question Type: Open-Ended **Response Format:** Free explanation required

↓

Meticulous transformation to constrain response format

Transformed Version

It's a hot rainy day and a brown hair man is standing on one side of a perennial river, with his dog on the other. There is only one option to cross the river — a bridge. It is heavily raining outside. The man calls his dog, who immediately crosses the river without getting a single fur wet and without using a bridge. On the other hand the man is completely wet. What is more likely: that the river is frozen or that the dog is bald?

Question Type: Closed-Ended **Response Format:** Binary choice (likelihood comparison)

Original Version

A man with curly hair goes out for a walk during a rainstorm with his wife. He doesn't have any protection of his own (umbrella, hood, cap, etc.), but his wife has an umbrella big enough for them both, which they make use of. By the end of his walk, there isn't a single wet hair on his head. Why doesn't the man have wet hair?

Question Type: Open-Ended **Response Format:** Free explanation required

↓

Meticulous transformation to constrain response format

Transformed Version

A man with curly hair goes out for a walk during a rainstorm with his wife. He doesn't have any protection of his own (umbrella, hood, cap, etc.), but his wife has an umbrella big enough for them both, which they make use of. By the end of his walk, there isn't a single wet hair on his head. What kept the man's hair dry?

Question Type: Closed-Ended **Response Format:** Specific object/entity identification

Figure 10: Transformation of open-ended riddles into closed-ended question formats. The scenarios remain identical while the question structures shift from requiring creative explanations to constraining responses through binary choices or specific entity identification.

B Metrics Results from Human Evaluations

As summarized in Fig 3, after three human evaluators independently annotated all 149 puzzles and we consolidated their judgments, o3 leads with 59.73% accuracy (89/149), followed by Gemini 2.5 Pro at 58.39% (87/149), Claude Sonnet 4 at 54.36% (81/149), Gemini 2.5 Flash at 48.32% (72/149), Claude 3.5 Sonnet at 46.31% (69/149), and GPT-4o at 29.53% (44/149).

Findings for CHATGPT (Figure 11). On the 149-puzzle set, *Thinking* attains 89/149 = 59.73% accuracy versus 44/149 = 29.53% for *Non-Thinking* (+30.2 pp). Agreements total 98/149 (both correct: 41; both wrong: 57). Disagreements strongly favor Thinking—48 Thinking-only vs 3 Non-Thinking-only—indicating substantial unique coverage from chain-of-thought.

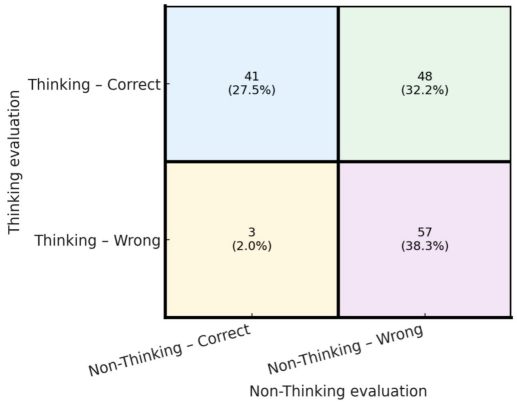


Figure 11: CHATGPT confusion matrix (n=149). Rows: Thinking outcome (Correct/Wrong); Columns: Non-Thinking outcome (Correct/Wrong). Counts and percentages are w.r.t. all puzzles. Agreement = 98/149 (65.8%): both correct 41 (27.5%), both wrong 57 (38.3%); disagreements favor Thinking—48 (32.2%) vs 3 (2.0%).

Findings for GEMINI (Figure 12). Thinking achieves 87/149 = 58.39% compared to 72/149 = 48.32% for Non-Thinking (+10.1 pp). Agreements are 98/149 (both correct: 54; both wrong: 44). Disagreements show 33 Thinking-only vs 18 Non-Thinking-only (net +15), indicating a consistent but smaller lift than for ChatGPT.

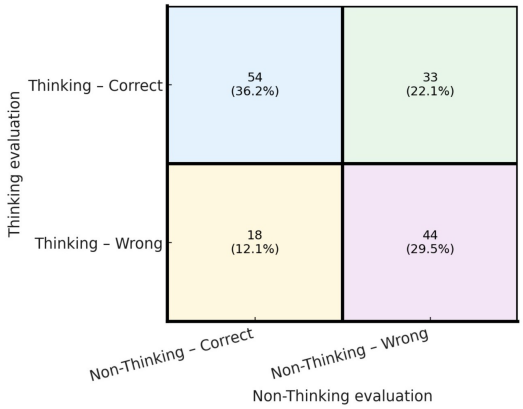


Figure 12: GEMINI confusion matrix (n=149). Rows: Thinking outcome (Correct/Wrong); Columns: Non-Thinking outcome (Correct/Wrong). Agreement = 98/149 (65.8%): both correct 54 (36.2%), both wrong 44 (29.5%); disagreements: Thinking-only 33 (22.1%) vs Non-Thinking-only 18 (12.1%).

Findings for CLAUDE (Figure 13). Thinking reaches 81/149 = 54.36% versus 69/149 = 46.31% for Non-Thinking (+8.0 pp). Agreements are lower at 87/149 (both correct: 44; both wrong: 43). Disagreements are 37 Thinking-only vs 25 Non-Thinking-only (net +12), suggesting the greatest complementarity between modes among the three models.

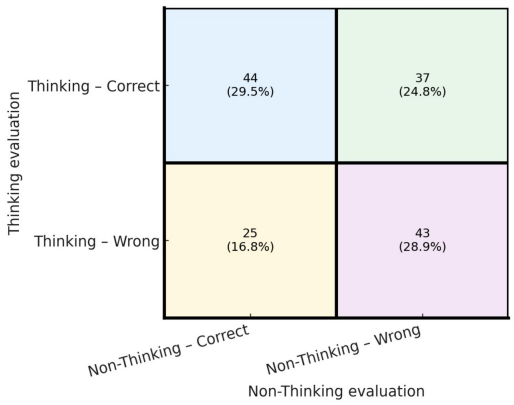


Figure 13: CLAUDE confusion matrix (n=149). Rows: Thinking outcome (Correct/Wrong); Columns: Non-Thinking outcome (Correct/Wrong). Agreement = 87/149 (58.4%): both correct 44 (29.5%), both wrong 43 (28.9%); disagreements: Thinking-only 37 (24.8%) vs Non-Thinking-only 25 (16.8%).

C Answer verification Auto-Evaluator

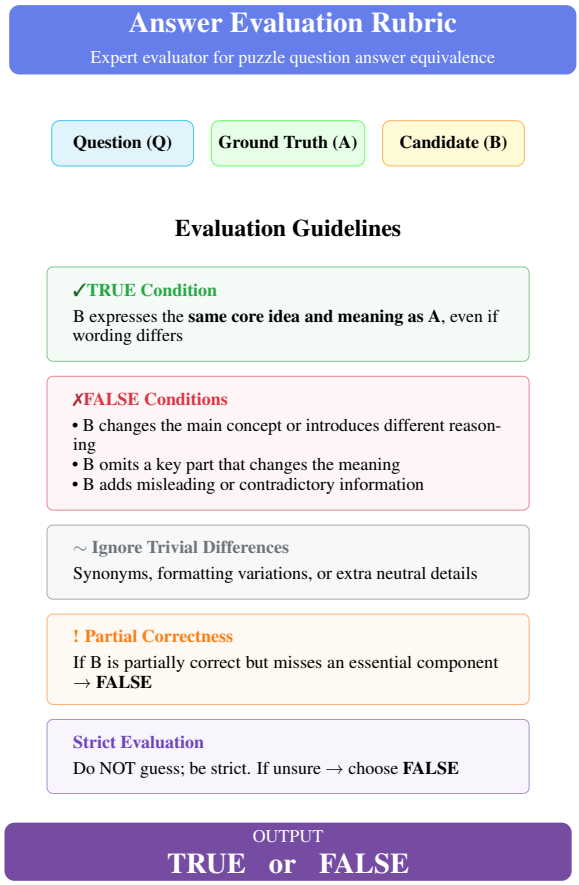


Figure 14: Evaluation rubric for assessing conceptual equivalence between ground truth and candidate answers in puzzle question tasks.

Fig 14 illustrates the rubric used by human annotators to judge whether a model's candidate answer (**B**) is *conceptually equivalent* to the ground-truth answer (**A**) for a given puzzle question (**Q**). Annotators prioritized *meaning preservation* over surface form: they marked TRUE only when **B** conveyed the same core idea and intended meaning as **A**, allowing for paraphrases, synonyms, formatting differences, or additional neutral details; they marked FALSE whenever **B** altered the claim (e.g., different or incorrect reasoning that changes the meaning), omitted an essential component, or introduced misleading/contradictory information, with *partial correctness* also treated as FALSE. The instructions were deliberately strict—if uncertain, annotators selected FALSE rather than guessing—yielding a binary decision (TRUE/FALSE) for each (**Q**, **A**, **B**) triple. Three independent human evaluators applied this rubric to all 149 items, and we consolidated their judgments into a single gold label per item, which we used for all accuracy analyses and figures in this paper.

Fig 15 illustrates how we apply the conceptual-equivalence criterion using two concrete cases. Each row shows the puzzle question (Q), the gold answer (A), and the model's candidate (B), followed by a green (TRUE) or red (FALSE) verdict. In *Example 1*, the candidate "*The dog is bald*" exactly matches the gold answer and thus preserves the intended meaning, so it is marked TRUE—note that the existence of another plausible story ("the river is frozen") is irrelevant because we judge *equivalence to A*, not general plausibility. In *Example 2*, the jugs puzzle already contains a jug with exactly 4L, so the gold answer is that the man can escape *immediately* (i.e., zero pours). The candidate "*1*" proposes a different outcome (one pour) and therefore changes the meaning; under our strict rubric (no partial credit for near misses), it is marked FALSE. These examples illustrate that judgments hinge on semantic identity with the gold answer rather than surface form or alternative—but non-gold—solutions.

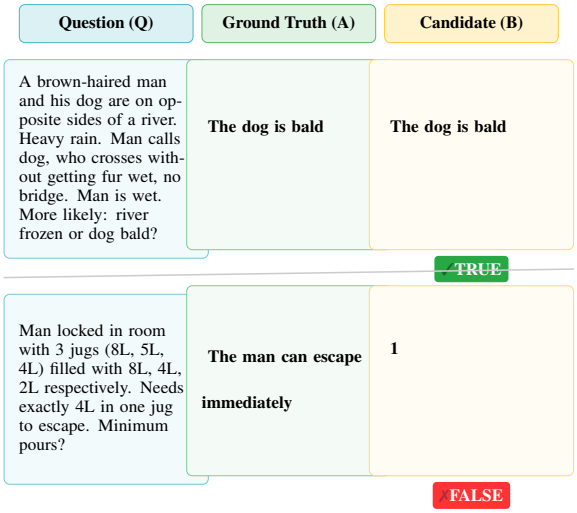


Figure 15: Examples of answer evaluation. **Example 1:** Candidate matches ground truth conceptually (TRUE). **Example 2:** Candidate answer "1" misses the key insight that one jug already contains 4L, making escape immediate without any pours (FALSE).

D Average Number of Steps Generated by LLMs

For incorrect answers, QWEN produces by far the longest chains of thought—51.1 steps on average (range 5–179), indicating a tendency to "over-reason" when it's off-track. In contrast, LLAMA is the most concise and consistent, with the lowest average of 5.0 steps and a tight 4–10 range. MISTRAL (7.8, 4–10) and INTERNLM (7.6, 5–13) show moderate length and variability, while PHI (8.8, 2–29) exhibits broader dispersion despite a modest mean. Overall, these results suggest that more steps do not equate to better outcomes—when models are wrong, longer reasoning often reflects digressions rather than recovery, whereas shorter, steadier traces (e.g., LLAMA) indicate more bounded but still unsuccessful reasoning. The information discussed is presented, for readability, in **Table 2**.

Table 2: Average number of reasoning steps generated by different LLMs for incorrect answers.

Model	Average Steps	Min Steps	Max Steps
MISTRAL	7.8	4	10
PHI	8.8	2	29
QWEN	51.1	5	179
LLAMA	5.0	4	10
INTERNLM	7.6	5	13

E Human Annotation Guidelines and Process

This appendix provides a comprehensive description of our human annotation methodology for identifying and categorizing errors in open source LLM-generated reasoning chains. The annotation process was carefully designed to ensure high-quality, reliable labels that serve as ground truth for evaluating our proposed error detection framework. We recruited three independent human evaluators for this task, each bringing complementary expertise to ensure comprehensive and nuanced error detection. The following structured guidelines were provided to all human evaluators and served as the primary reference throughout the annotation process:

Step-by-Step Annotation Protocol

1. Take a reasoning chain that has been broken down into steps
2. Break each step into premise and conclusion
3. Classify the step into one of the **5 Main Categories**:
   - **Logically Sound Step:**Correct conclusion validly derived from correct premise
   - **Deductive Failure:**Correct premise leads to incorrect conclusion due to flawed reasoning
   - **Compounded Failure:**Incorrect premise leads to incorrect conclusion
   - **Fortuitous Correctness:**Flawed premise coincidentally results in correct conclusion
   - **Declarative Statement:**Restates a clue or previous finding without drawing new conclusion
4. If errors exist, identify the **Premise level Error** (if any):
   - Evidence Misrepresentation
   - Insufficient Evidence
   - Unsupported Assumption
   - Cascading Error
   - Faulty Postulate
5. If errors exist, identify the **Conclusion level Error** (if any):
   - Consequential Error
   - Improper Candidate Elimination
   - General Logical Fallacy

Important Notes:

- A single reasoning step may contain multiple errors at the same level or across both levels
- When uncertain, err on the side of marking an error if the reasoning appears questionable or insufficiently justified
- *Cascading Error* should be marked only when the current step's premise directly inherits an error from a previous step's faulty conclusion
- Provide brief justification for each annotation decision in the notes field
- Consider both explicit and implicit assumptions when evaluating premises

The pairwise coincidence (percent agreement) among the three human evaluators is $A_{12} = 90\%$, $A_{13} = 93\%$, and $A_{23} = 89\%$. The average pairwise coincidence is therefore $\bar{A} = \frac{A_{12}+A_{13}+A_{23}}{3} = \frac{90+93+89}{3} = 90.67\%$ (approximately 90.7%), indicating high consistency across annotators (range 89%–93%).

F The LLM-based Step Error Classification Autoevalutor

This appendix describes the architecture and implementation of our LLM-based auto-evaluator for reasoning step analysis.

F.1 Architecture Overview

Our auto-evaluator employs a two-agent pipeline to systematically analyze each reasoning step (Figure 16). Each agent performs a specialized function:

1. **Deconstructor:** Parses a reasoning step into its foundational premise and resulting conclusion.
2. **Validator:** Evaluates the correctness of both the premise (against context) and the conclusion (given the premise).
3. **Classifier:** Maps the validation results to our error taxonomy.

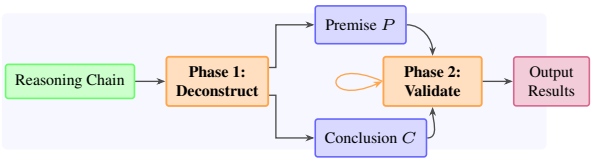


Figure 16: Two-phase reasoning validation pipeline. Phase 1 extracts premise and conclusion from the reasoning chain, which Phase 2 then validates with iterative refinement (loop).

F.2 Agent Prompts

The Agent Prompts section provides the complete prompt specifications for each component of our two-phase auto-evaluator system. As described earlier, the auto-evaluator employs a two-agent pipeline to systematically analyze reasoning chain steps, with each agent performing a specialized function in the evaluation process. The prompts detailed in this section define the specific instructions, input formats, and expected outputs for the Deconstructor, Validator, and Classifier agents, ensuring that the evaluation process is both rigorous and replicable.

The Deconstructor prompt (Figure 17) guides the agent to parse each reasoning step into its foundational premise and resulting conclusion, establishing the atomic units for subsequent evaluation. This prompt instructs the model to identify the key information being assumed or referenced (the premise) and the logical inference or claim being made (the conclusion), providing structured output that facilitates downstream validation. The prompt explicitly defines the distinction between premises and conclusions, providing examples of each to ensure consistent interpretation across diverse reasoning steps. Additionally, the Deconstructor is instructed to preserve the semantic content of the original step while reformulating it into the premise-conclusion structure, maintaining fidelity to the model's original reasoning process.

The Validator prompt (Figure 18) specifies how to assess the correctness of both the extracted premise (against the provided context) and the conclusion (given the premise), producing structured validation results. The prompt defines clear criteria for evaluating premise validity—whether the assumed information is supported by the given context—and conclusion soundness—whether the inference logically follows from the premise. The validator produces binary judgments for each component along with justifications for its assessments. Critically, the prompt emphasizes that premise validation should be performed strictly against the provided context without incorporating external knowledge, while conclusion validation should assess logical entailment independent of whether the premise itself is correct. This separation allows us to distinguish between errors stemming from faulty assumptions versus errors in logical inference. The prompt also includes detailed instructions for handling edge cases, such as implicit premises, multi-part conclusions, and reasoning steps that draw on multiple contextual elements.

Finally, the Classifier prompt (Figure 19) maps these validation outcomes to our error taxonomy, enabling systematic categorization of reasoning failures. Based on the premise and conclusion validity from the Validator, the Classifier assigns each erroneous step to one of our predefined error categories. The prompt provides the complete error taxonomy with definitions and examples, ensuring classification decisions are grounded in our theoretical framework. Together, these prompts form the operational backbone of our automated evaluation framework, ensuring consistent and reproducible analysis across all reasoning chains in our dataset.

Agent 1: Deconstructor

You are a precise logical analysis tool. Your task is to deconstruct a reasoning step into its foundational premise and its resulting conclusion.
Context:
`{context}`
Reasoning Step:
`{reasoning_step}`
Based on the reasoning step, identify the premise (the starting fact or assumption) and the conclusion (the inference drawn from it). Provide your answer in JSON format only, with the keys "premise" and "conclusion". Do not include any explanatory text or markdown formatting.

Figure 17: Agent 1 (Deconstructor): Deconstructs reasoning steps into premise and conclusion components.

Agent 2: Validator

You are a meticulous logical validator. You will receive a context, a premise, and a conclusion. Evaluate each part and provide a structured JSON response.
Error Sub-categories:

For wrong premises: (1) Evidence Misrepresentation (2) Insufficient Evidence (3) Unsupported Assumption (4) Cascading Error (5) Faulty Postulate
For wrong conclusions: (a) Consequential Error (b) Improper Candidate Elimination (c) General Logical Fallacy
Context:
`{context}`

Analysis Input:
Premise: `{premise}`
Conclusion: `{conclusion}`
Instructions:

1. **Evaluate the Premise:** Compare the premise ONLY against the provided context. Is it factually correct according to the context? If it's wrong, which sub-category best explains why? Determine if its status is "Right Premise" or "Wrong Premise".
2. **Evaluate the Conclusion:** Assuming the premise is TRUE, does the conclusion logically follow from it? If the reasoning is flawed, which sub-category best explains why? Determine if its status is "Right Conclusion" or "Wrong Conclusion".

Provide your analysis in the following JSON format ONLY. Your entire response must be a single, valid JSON object, starting with { and ending with }.

```
{
  "premise_evaluation": {
    "status": "Right Premise OR Wrong Premise",


    "reason_code": "1-5 or N/A",
    "analysis": "Briefly explain your reasoning."
  },
  "conclusion_evaluation": {
    "status": "Right Conclusion OR Wrong Conclusion",


    "reason_code": "a-c or N/A",
    "analysis": "Briefly explain if the conclusion follows."
  }
}
```

Figure 18: Agent 2 (Validator): Validates premise against context and evaluates logical consistency of conclusions.

Agent 3: Classifier

You are an error classification system. Convert the provided validation report into a final error code based on the given taxonomy. Respond with the final code only.
Taxonomy:

Compounded Failure: Wrong Premise & Wrong Conclusion
Fortuitous Correctness: Wrong Premise, Right Conclusion
Deductive Failure: Right Premise, Wrong Conclusion
Logically Sound Step: Right Premise, Right Conclusion
Declarative Statement: No Conclusion / pure restatement

Validation Report:
`{report_str}`
Instructions:
1. Determine the broad category (Compounded Failure, Fortuitous Correctness, Deductive Failure, Logically Sound Step, Declarative Statement) from the status fields.

2. Append the reason codes in parentheses.
3. If the premise is right, omit its reason code.
4. If the conclusion is right, omit its reason code.
5. Format: Category(PremiseCode, ConclusionCode) or Category(Code).
Respond with only the final code.

Figure 19: Agent 3 (Classifier): Classifies validation results into final error taxonomy codes.

G Effectiveness of Error Mitigation

The effectiveness of our error mitigation strategy is demonstrated through the design and implementation of a carefully structured prompt template that embodies the principles of explicit prohibitory instructions. Our prompt engineering process was characterized by **meticulous iterative refinement and strengthening**, where each component was systematically tested, evaluated, and enhanced based on observed failure modes across diverse problem sets.

G.1 Prompt Architecture and Design Rationale

The final prompt template employs a multi-layered architecture that guides the model through a structured problem-solving protocol while simultaneously imposing strict constraints on its behavior. The template is organized into three primary phases: **Phase 1: Understanding (Step 1 - UNDERSTAND).** This initial phase explicitly instructs the model to engage in careful problem comprehension by reading the problem twice, identifying the specific question being asked, and enumerating all given information. This deliberate repetition serves to anchor the model's attention on the problem statement itself rather than pattern-matching to similar problems from training data. **Phase 2: Solution Generation (Step 2 - SOLVE).** The solving phase emphasizes transparency and incrementality, requiring the model to work step-by-step, show all calculations and logical reasoning, and perform inline verification at each step. This structured approach reduces the likelihood of logical leaps or omitted reasoning that could introduce errors. **Phase 3: Verification (Step 3 - VERIFY).** The final phase implements a systematic checking mechanism with three specific validation criteria: confirming the answer addresses the actual question posed, verifying correct units and format, and validating the solution by substitution back into the original problem.

G.2 Strategic Use of Prohibitory Instructions

A distinguishing feature of our approach is the **STRICT RULES** section, which explicitly delineates prohibited behaviors using visual markers (✗) for constraints and approved behaviors (✓) for acceptable fallback strategies. The prohibitions specifically target the most common failure modes observed during development:

- **External knowledge restriction:** "NO external knowledge or memorized solutions" prevents the model from applying domain-specific facts not present in the problem statement, directly addressing hallucination tendencies.
- **Assumption prevention:** "NO assumptions about unstated information" and "NO assuming 'standard' interpretations unless explicitly stated" combat the model's tendency to fill gaps with plausible but unwarranted information.
- **Procedural rigor:** "NO skipping steps in calculations" enforces complete working, making errors more detectable and reasoning more transparent.
- **Explicit fallback mechanism:** The instruction to state "Cannot solve: missing [X]" when information is ambiguous provides the model with a legitimate alternative to guessing, reducing false confidence in incomplete solutions.

G.3 Iterative Refinement Process

The development of this prompt template involved systematic strengthening through multiple iterations. Initial versions used general instructions like "solve carefully" or "show your work", which proved insufficient for error mitigation. Through empirical testing on benchmark problems, we identified specific failure patterns—such as models assuming standard values (e.g., gravitational acceleration) without being told, or answering related but different questions than what was asked. Each identified failure mode prompted the addition or strengthening of a corresponding constraint. For example, early versions lacked the "re-read the original question" instruction in the CRITICAL section, leading to instances where models would solve a problem correctly but report an answer in the wrong format or for a subtly different question. The addition of this explicit re-verification step, with its three-point checklist, reduced such format and question-mismatch errors substantially. The visual distinction between prohibited (✗) and approved (✓) behaviors was introduced after observing that purely textual negative constraints were sometimes overlooked. The use of visual markers increased the salience of these instructions, improving adherence rates.

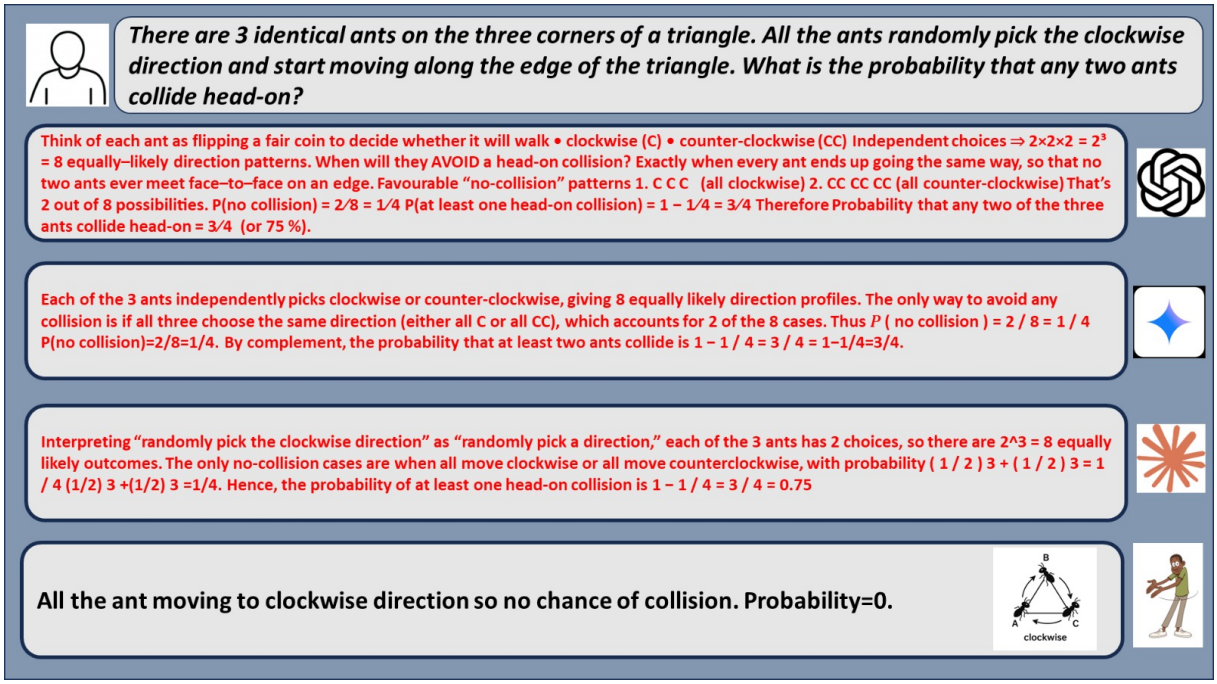
G.4 Performance Impact

As noted previously, this prompt methodology achieved approximately **10% improvement on closed-source LLMs** and **just over 5% improvement on open-source LLMs** compared to baseline prompting approaches. The performance gains were particularly pronounced in problems requiring:

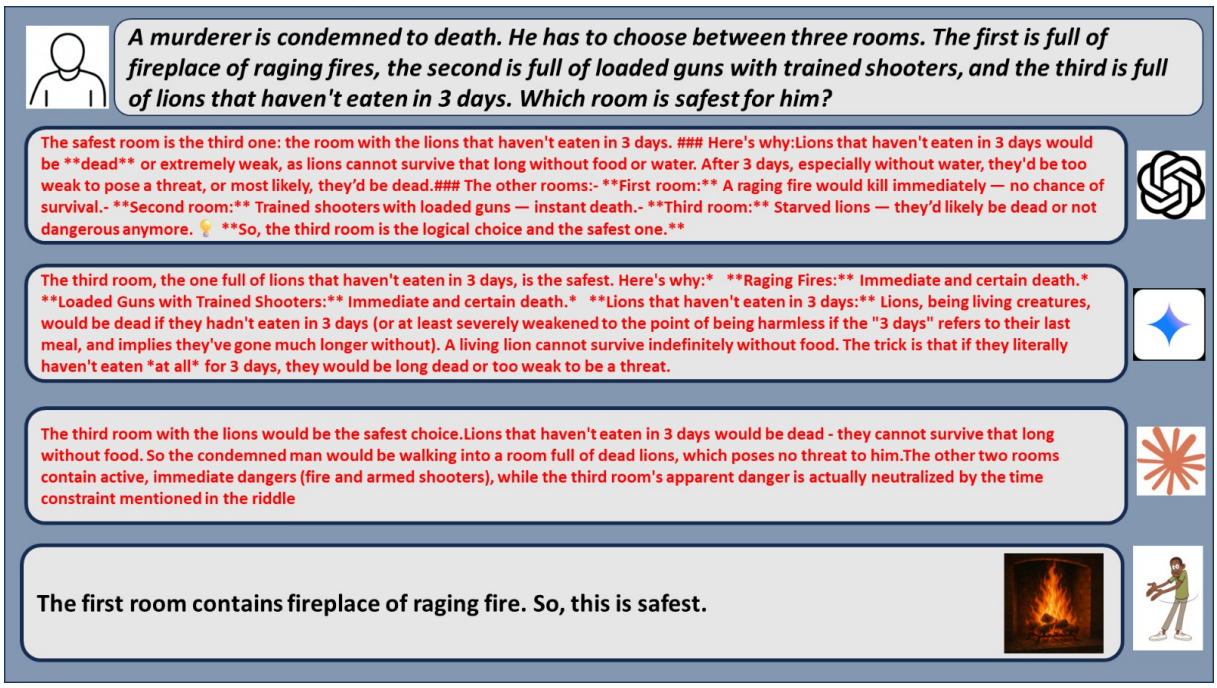
- Careful distinction between given and assumed information
- Multi-step reasoning with intermediate verification
- Precise adherence to output format specifications
- Recognition of insufficient information to solve a problem

Notably, the structured three-phase protocol combined with explicit prohibitions proved more effective than either component alone, suggesting a synergistic effect between positive guidance (the step-by-step structure) and negative constraints (the prohibitory rules). The final "CRITICAL" section, which forces a terminal verification pass before finalizing the answer, served as a crucial safety net that caught errors that survived earlier checking phases. This comprehensive approach demonstrates that effective error mitigation in LLMs requires not merely instructing models on what to do, but equally importantly, explicitly constraining what they should *not* do, coupled with structured protocols that enforce verification at multiple stages of the problem-solving process.

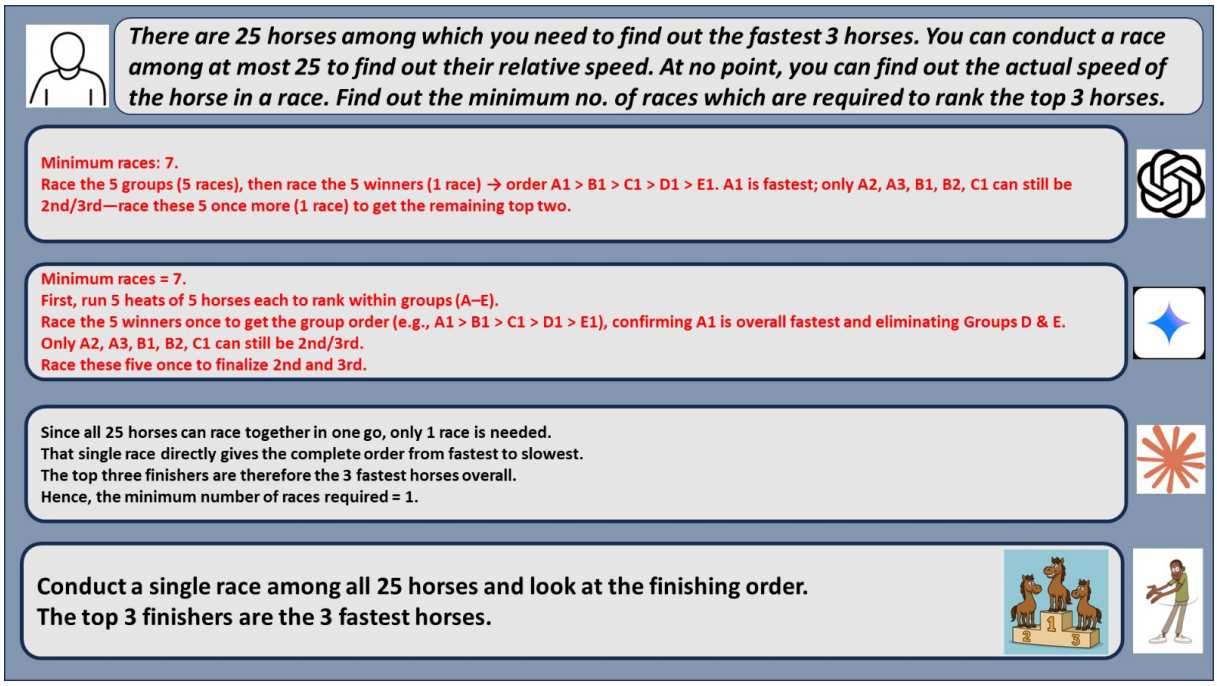
H Some examples of puzzles where most of the closed source models fail



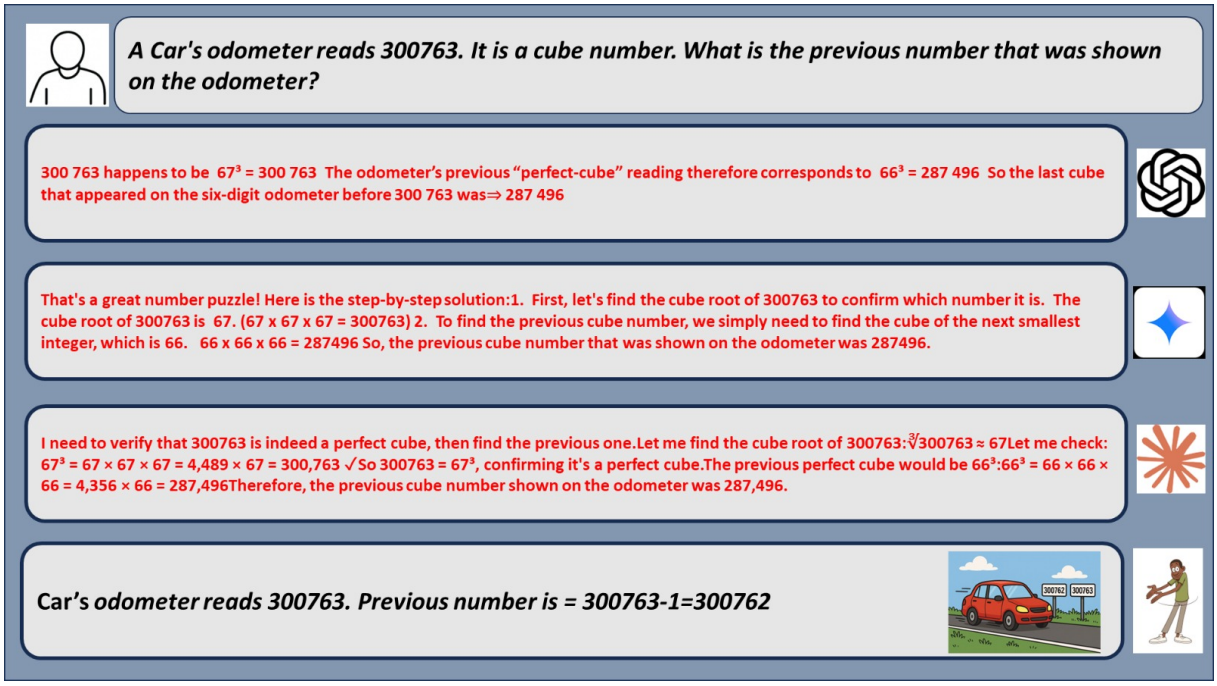
(a) Puzzle variation where GPT, Gemini, and Claude fail.



(b) Puzzle variation where GPT, Gemini, and Claude fail.



(c) Variation where GPT, Gemini fail but Claude succeeds.



(d) Variation where all three models fail.

Figure 20: Examples of puzzle variations from our dataset.