ROBUST REASONING BENCHMARK

Anonymous authors
Paper under double-blind review

ABSTRACT

While Large Language Models (LLMs) achieve high performance on standard mathematical benchmarks, their problem-solving ability remains sensitive to context and textual formatting. We introduce the *Robust Reasoning Benchmark (RRB)*, a pipeline of 13 deterministic textual perturbations applied to AIME 2024 and AIME 2025. Evaluating 8 state-of-the-art models, we find that frontier models are largely resilient, with the notable exception of Claude 4.6 Opus, which categorically refuses many transformed prompts, while open-weights reasoning models degrade by up to 53% on average and up to 100% on individual perturbations. By prompting models to solve several unperturbed problems in sequence, we identify *Intra-Query Attention Dilution*: open-weights models from 7B to 120B parameters lose accuracy on later problems. At matched preceding token length, the same models perform better when earlier reasoning is replaced with neutral text. Per-layer attention traces show persistent attention mass on the earlier problems.

1 INTRODUCTION

Large Language Models (LLMs) have achieved remarkable success on mathematical reasoning benchmarks, with state-of-the-art models approaching saturation on datasets such as AIME, GSM8K, and MATH (Cobbe et al., 2021; Hendrycks et al., 2021; Mathematical Association of America, 2024). These results have fueled the perception that LLMs possess robust algorithmic reasoning capabilities: the ability to decompose complex problems into logical intermediate steps. However, a growing body of literature (McCoy et al., 2023; Jiang et al., 2024; Nezhurina et al., 2025) suggests that this performance may be brittle, relying instead on probabilistic pattern matching and surface-level correlations.

To distinguish between robust reasoning and fragile memorization, recent research has turned to adversarial evaluation. Existing approaches generally fall into two categories: changing numeric values to test arithmetic generalization (Qian et al., 2024; Yang et al., 2025; Sun et al., 2025a) or increasing environmental complexity by adding irrelevant context or harder logic (Shi et al., 2023; Jiang et al., 2025; Anantheswaran et al., 2025). While valuable, these methods often introduce confounding variables. When a model fails after numerical perturbation, it is unclear whether the failure lies in the reasoning logic or the arithmetic calculation. Similarly, when a model fails on a more complex problem, it is difficult to distinguish between a lack of robustness and a hard limit on the model's inherent capacity. Furthermore, some benchmarks rely on LLMs to generate the perturbations, introducing non-deterministic noise and potential validity errors into the evaluation pipeline.

In this paper, we introduce the **Robust Reasoning Benchmark (RRB)**, an evaluation framework based on 13 deterministic structural transformations. Unlike prior work, we do not alter the mathematical values, the logic, or the final answer. Specifically, we apply reversible algorithmic perturbations such as syntactic string reversals, visual 2D grid encodings (e.g., Rail Fence, Horizontal Snake (Wikipedia, 2026; Merriam-Webster, 2026)), and semantic wrappers. We provide a description of each transformation in context, isolating the ability to invert a transformation from the ability to recognize it. The transformations are mechanically simple to invert but structurally adversarial to LLM tokenization and attention. Frontier models are largely resilient, with the notable exception of Claude 4.6 Opus, which categorically refuses many transformed prompts; open-weights reasoning models lose up to 53% accuracy on average across transformations.

We also measure whether models can recover a transformed statement when solving is removed from the task. In separate runs on line-interleaved problems, Qwen3-30B reconstructs the statement in

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

User Prompt Example

```
Solve these unrelated problems.
Put each answer in \boxed{}.
Problem 1:  Find the number ...
Problem 2:  Let ABC be a ...
Problem 3:  Let p be the ...
```

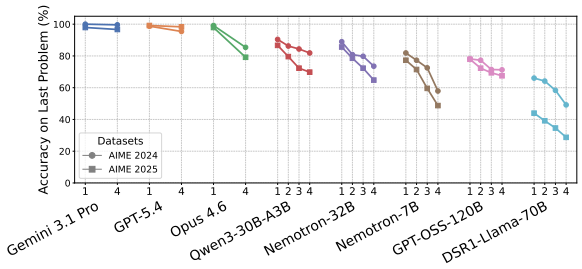


Figure 1: **Intra-Query Attention Dilution.** *Left:* The multi-problem task, in which models solve multiple independent problems within a single prompt. *Right:* Accuracy on the *final* problem of the sequence, against the number of problems in the sequence. Proprietary models were sampled only at 1 and 4 problems because of API cost. While frontier APIs are more resilient, every open-weights model loses accuracy as problems accumulate. The comparison with matched passive context appears in Table 2.

93% of samples and solves the transformed problem in 77%. These are rates for different tasks and trajectories; they do not show which solve failures followed successful decoding. They show that the transformed statement can often be reconstructed when reconstruction is the only requested output.

In a separate experiment, we remove the transformations and ask models to solve several unperturbed problems within a single query. We measure accuracy only on the last, identifying **Intra-Query Attention Dilution**: as visualized in Figure 1, accuracy on the final problem degrades as the context fills with the model's own prior reasoning. A control in which that prior reasoning is replaced by neutral text of identical preceding token length yields higher target accuracy (Table 2). Long-context attention drift is a recognized challenge in information retrieval (Salvatore et al., 2025; Chroma Research, 2025); here it arises within a single query, over text the model generated itself. These observations motivate testing whether a model can reason more reliably when completed steps are summarized or removed from its active context. Approaches that delegate steps to sub-agents (Ruan et al., 2026; Zhang et al., 2026) provide one way to study such boundaries. Appendix E discusses how the size of each reasoning step could affect this approach.

In summary, our contributions are:

- A set of 13 deterministic, reversible textual transformations that alter the presentation of a mathematical problem while preserving its content.
- The *Robust Reasoning Benchmark (RRB)*, which applies those transformations to AIME 2024 and AIME 2025 and evaluates five open-weights and three proprietary models.
- Measurement of *Intra-Query Attention Dilution* in the multi-problem task, with an equal-length passive-context control separating self-generated reasoning from context length, and mechanistic attention analysis showing that prior reasoning steps retain attention mass.

2 Background and Related Work

Researchers have introduced several benchmarks to evaluate the performance of LLMs on mathematical reasoning tasks, such as GSM8K, MATH, and AIME 2024 (Cobbe et al., 2021; Hendrycks et al., 2021; Mathematical Association of America, 2024). Techniques such as Chain-of-Thought (CoT) and Tree-of-Thought (ToT) (Wei et al., 2022; Yao et al., 2023) have been the primary drivers of this success. However, static benchmark accuracy can mask the fragility of the underlying thinking process, and distinguishing genuine reasoning from pattern matching remains a challenge. A critical attribute of robust reasoning is invariance to surface-level perturbations.

Perturbing the problem statement. A prominent line of work evaluates robustness by altering the input. One group introduces irrelevant information or lexical shifts: Mirzadeh et al. (2025), Anantheswaran et al. (2025), and others (Shi et al., 2023; Jiang et al., 2024; 2025) inject irrelevant numeric variables or token biases, while Wang & Zhao (2024), Zhao et al. (2025), and meaning-preserving rephrasing studies (Kirtane et al., 2025; Hao et al., 2025; Sun et al., 2025b; Karim et al., 2025; Hou et al., 2025; Zhu et al., 2024; Alavi et al., 2025) quantify how homophones, sentence

Table 1: The 13 transformations. *Unit* is the granularity at which the transformation operates. All transformations are deterministic and programmatically invertible. Except for Not Not, which requires no decoding, we supply the rule in plain English (Appendix A).

Category	Transformation	Unit	Rule
Semantic & Lexical	Not Not	token	Insert double negations before values and adjectives
	Opposites	token	Remap terms to antonyms; key defined in-query
	Wrappers	token	Wrap terms in identity functions, e.g. `3(morning)`
Contextual Overload	Interleave (Line)	line	Alternate 60-character tagged segments of two problems
	Interleave (Word)	word	Alternate the words of two problems
	Interleave (Symbol)	character	Alternate the characters of two problems
Syntactic Distortions	Sentence Reversal	sentence	Reverse the order of sentences
	Word Reversal	word	Reverse the order of words
	Symbol Reversal	character	Reverse the characters within every word
Visual & Spatial	Snake Horizontal	character	Write into a grid in boustrophedon row order
	Rectangle Perimeter	character	Write clockwise around a rectangle's edges
	Snake Vertical	character	Write into a grid in zigzag column order
	Rail Fence	character	Zigzag the characters across multiple rails

rephrasings, and visual attacks degrade accuracy. A second group alters values and variables to test whether the model reproduces memorized training data (Qian et al., 2024; Nezhurina et al., 2025; Li et al., 2024; Sun et al., 2025a; Xie et al., 2024), or converts static problems into code functions to enable automatic generation and measure memorization limits (Srivastava et al., 2024; Xie et al., 2025); McCoy et al. (2023) show that performance depends strongly on text probability. A third group increases the difficulty of existing benchmarks (Ito et al., 2025; Shalyt et al., 2025; Shojaee et al., 2025; Li et al., 2026; Huang et al., 2025; Hong et al., 2025) through controlled environments, confusing parameter substitutions, or mathematically harder rewrites. Each of these modifies the task itself: changing numbers alters the mathematical constraints, added difficulty alters the capacity required, and LLM-generated rephrasings introduce non-determinism and semantic ambiguity. Our perturbations are strictly deterministic, operate purely at the textual level, and preserve the exact mathematical state; by supplying the decoding rule we isolate structural fragility without altering the problem's semantic or logical constraints. Unlike McCoy et al. (2023), which tests text decoding alone, we require the model to decode *and* reason, and unlike the difficulty-based benchmarks our transformations hold difficulty constant. The method is also domain-agnostic and applies to any LaTeX-formatted mathematics dataset without manual rewriting.

Perturbing the reasoning trajectory. A smaller line of work perturbs the intermediate steps of a model's reasoning rather than its input. Work by von Recum et al. (2026) and Aravindan & Kejriwal (2026) injects errors or interventions directly into the CoT, showing that internal reasoning processes remain susceptible to derailment. Our second set of experiments is most closely related to these. Our multi-problem task uses the model's own prior chain-of-thought, accumulated within a single query, and compares it with neutral text of matched preceding token length in Section 5.3.

3 The Robust Reasoning Benchmark

RRB changes the text of a problem without changing its mathematical premises or answer. We use 13 deterministic transformations in four groups. Table 1 lists each rule and the unit of text it changes. Figure 2 shows what the transformed prompts look like.

The semantic and lexical rules change how familiar terms appear. *Not Not* inserts double negations. *Opposites* replaces selected words with antonyms, while *Wrappers* encloses terms in identity functions. The latter two prompts include a `defyn{...}` block that defines the substitutions. The model can recover the original terms from that key without guessing their meanings.

The interleaving rules combine a target problem with an unrelated problem. They alternate tagged 60-character segments, words, or individual characters from the two statements. The instruction asks the model to solve only the target. If one statement ends first, its text repeats from the beginning

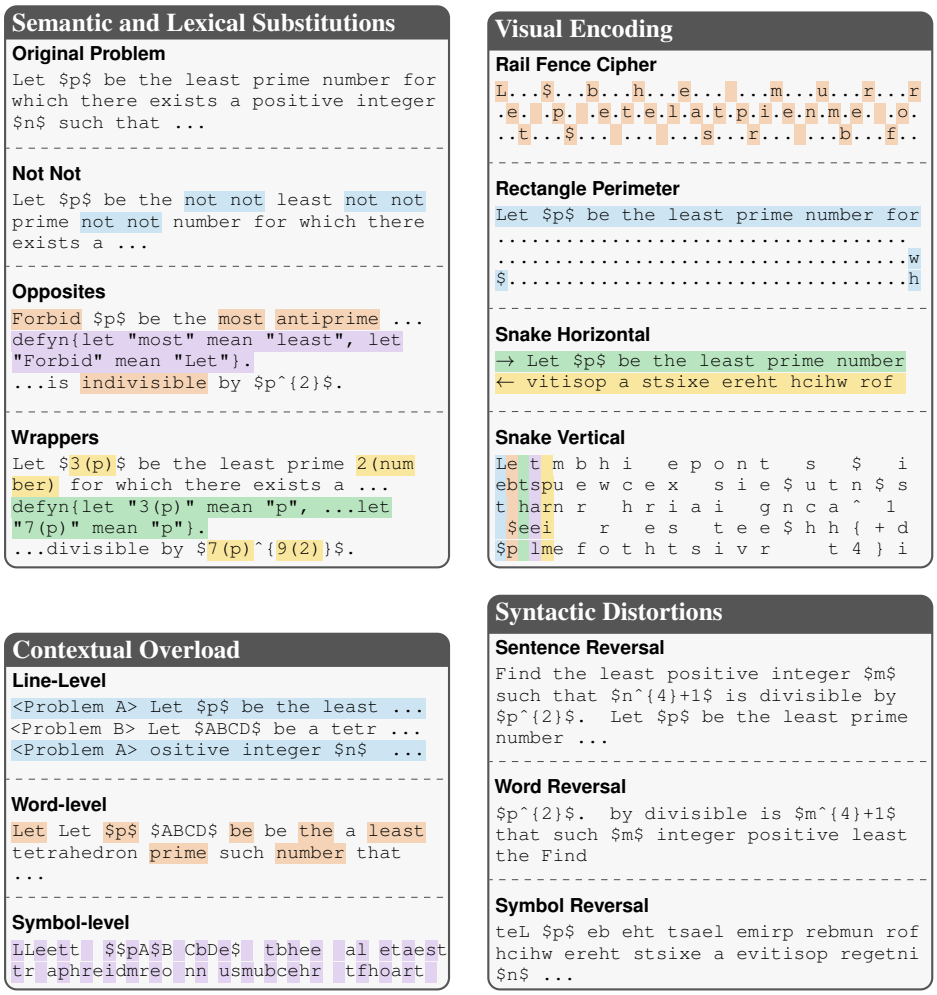


Figure 2: Examples of the 13 transformations, grouped by category. Within each group, the examples progress from larger text units to smaller ones; Table 1 lists the unit for each rule. Long statements and mappings are shortened with ....

to fill the remaining positions. Sentence Reversal changes the order of sentences; Word Reversal changes the order of words; and Symbol Reversal reverses the characters within each word.

The four visual rules lay out the statement as a grid. Snake Horizontal alternates left-to-right and right-to-left rows, while Snake Vertical alternates downward and upward columns. Rectangle Perimeter writes the statement clockwise around a rectangle, and Rail Fence places successive characters on zigzagging rows. The prompts mark the start and end of each grid and use dots for empty cells so that the positions remain visible.

These rules operate at different *granularities*, from whole sentences to individual characters. We compare results across granularities in Section 5.2. Each transformation has a deterministic inverse that we use to check the generated input.

4 Methodology

We apply the transformations in Section 3 to AIME 2024 (Mathematical Association of America, 2024) and AIME 2025 (Mathematical Association of America, 2025). Before transforming a problem, we remove LaTeX comments, replace each line break with a semicolon and a space, and insert a space after any of the letters `b`, `n`, `t`, `a`, `f`, or `r` when immediately followed by a backslash. This

prevents reversal from creating control sequences such as `\b`. We apply the same preprocessing to the untransformed baseline. Appendix G gives an example and the exact rules.

Each transformation leaves the mathematical premises and answer intact. We verify that a deterministic inverse recovers the preprocessed statement, without using a language model or a heuristic parser. For each rule that requires inversion, the model receives the decoding instructions in plain English. The system and user instruction templates appear in Figure 8.

We measure problem-solving accuracy by extracting answers in the `\boxed{}` format with the `math_verify` package (Face, 2026) and checking them against the reference answer. We count outputs that reach the token limit as failures. We also record the number of output tokens; Appendix D analyzes this measure. We use 16 zero-shot trajectories per problem for the open-weights models and 8 for the proprietary APIs.

To test what happens when a model must continue reasoning after finishing a problem, we ask it to solve several unrelated, untransformed problems in one query. We score only the last problem. We search all `\boxed{}` blocks for its answer because some models answer the problems out of order. The single-problem condition provides the baseline.

We run two controls. In the decode-only control, the model receives the same transformed input and decoding rule, but reconstructs the original statement instead of solving it. We compare its reconstruction with the oracle inverse after normalizing LaTeX presentation and whitespace. A reconstruction passes if it matches exactly or has character error rate (CER) at most 0.02. We compute CER as $(S + D + I)/N$, using substitutions, deletions, and insertions over the N reference characters with `jiwer` (Vaessen, 2026). We omit *Not Not*, which requires no reconstruction, and sample trajectories at the same rate as in the solve condition. Some outputs contain the requested tags but little or no problem text. We report the *stub rate*, the fraction shorter than one quarter of the oracle statement, alongside the recovery rate so readers can distinguish non-attempts from substantive reconstructions.

The passive-context control compares the effect of prior reasoning with the effect of context length. For each sequential-problem sample, we locate where the model begins the final problem. We then create a separate prompt with the same target and a contiguous, non-repeating passage of neutral prose. The passage is chosen so that the number of tokens before the target matches that sample. This removes the earlier mathematical problems and their generated solutions while holding the preceding token count fixed. We grade the target in the same way in both conditions.

We also measure attention while three earlier problems and a fourth target problem are solved in one query. We divide the sequence into instruction tokens, earlier problems with their generated reasoning, and the target with its generated reasoning. For each token generated while solving the target, we sum the attention mass assigned to each region and average over heads and target tokens at each layer. Appendix H defines the measurement formally.

Our evaluation covers five open-weights models: Qwen3-30B-A3B-Thinking (Qwen Team, 2026), OpenReasoning-Nemotron-7B (NVIDIA, 2025b), OpenReasoning-Nemotron-32B (NVIDIA, 2025a), DSR1-Llama-70B (DeepSeek-AI, 2025), and gpt-oss-120b (OpenAI, 2025). The three proprietary models are Gemini 3.1 Pro (Google DeepMind, 2026), GPT-5.4 (OpenAI, 2026), and Claude 4.6 Opus (Anthropic, 2026). The decode-only and passive-context controls cover Qwen3-30B-A3B and both Nemotron models on both datasets. We ran the sequential-problem task on proprietary models only with one and four problems. Attention extraction covers these three open-weights models; the 70B and 120B models exceeded our hardware limits for that analysis. Appendix B lists checkpoints, decoding settings, and hardware.

We treat the AIME problem, rather than each trajectory, as the unit of uncertainty because trajectories on the same problem are not independent. We report bootstrap intervals over the 30 problems for each condition, using 10,000 resamples. For paired comparisons, we resample problem-level differences.

5 Results and Discussion

We evaluate accuracy under 13 input transformations and when models solve several independent problems in one query.

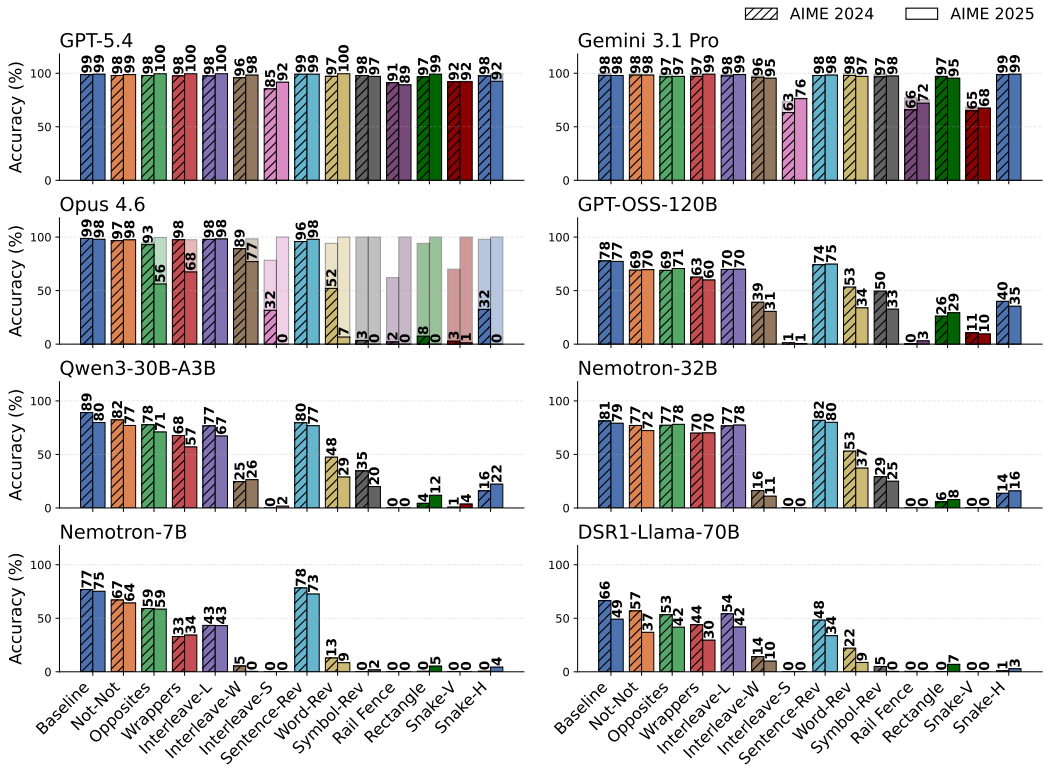


Figure 3: Accuracy on AIME 2024 and AIME 2025 under each of the 13 structural perturbations. Solid bars show accuracy; semi-transparent segments indicate refusals or failed answers.

5.1 Robustness to Perturbations: Frontier and Open-Weights Models

Accuracy under textual perturbation differs by model class (Figures 4 and 3). GPT-5.4 and Gemini 3.1 Pro degrade by only 3% and 7% on average, while open-weights models with comparable baseline accuracy lose 53% for Nemotron-7B and 47% for Qwen3-30B-A3B-Thinking-2507. With the AIME problem as the unit of analysis, 149 of the 208 (model, dataset, transformation) cells have bootstrap intervals that exclude zero. Truncation does not account for these drops: correcting for it changes accuracy by less than 1 point in 170 of 208 cells (Appendix J).

The largest drops for open-weights models occur on *Syntactic Distortions* and *Visual Encodings*, while semantic substitutions have smaller effects. Transformations that isolate characters, such as *Interleave (Symbol)* and *Rail Fence*, break up familiar subword units. We call this input fragmentation *tokenization breakdown*. Qwen3 and DSR1 score at most 1.7% on these transformations. Orientation also matters: on AIME 2024, Gemini 3.1 Pro scores 99% on *Snake (Horizontal)* but 65% on *Snake (Vertical)*, and Qwen3 16% versus under 1%. This is consistent with a bias toward left-to-right presentation. Another failure mode, *cognitive thrashing*, appears in token consumption rather than accuracy: instead of terminating, open-weights reasoning models enter long unproductive loops, generating up to 27K tokens while achieving 0% accuracy (Appendix D).

Overrefusal. Claude 4.6 Opus is a notable exception among the closed-source models, dropping 52% on average. This drop reflects refusal rather than reasoning failure: the model declines most prompts involving symbol manipulation (the semi-transparent segments in Figure 3). We hypothesize that safety filtering treats these symbol-manipulation instructions as prompt injection.

5.2 Decode-Only Recovery

We repeat the perturbation experiment with the solving step removed (Section 4). The model must reconstruct the entire original statement to a character error rate of at most 0.02. The median statement has 255 non-whitespace characters, so the criterion tolerates about five errors. Solving, by

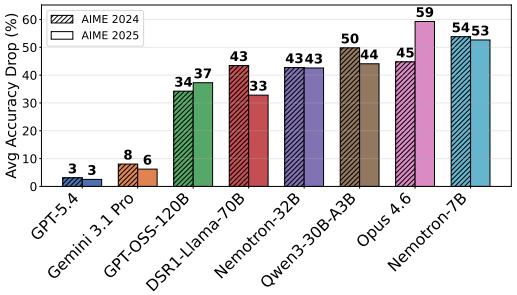


Figure 4: Average accuracy drop per model across the 13 structural perturbations on AIME 2024 and AIME 2025.

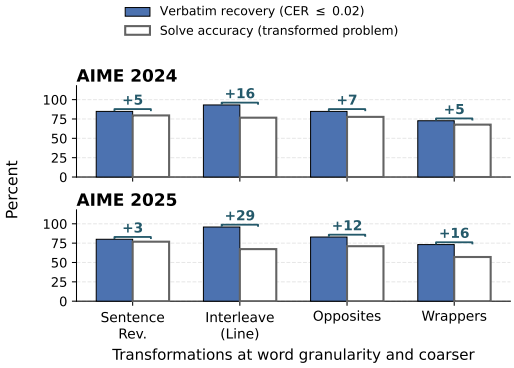


Figure 5: Decode-only recovery and solve accuracy for Qwen3-30B-A3B on four coarse transformations. Brackets label the percentage-point difference between rates measured in separate runs. See Appendix I for full results.

contrast, requires a single integer. These tasks have different output demands, especially when the transformation scrambles individual characters.

Figure 5 therefore focuses on four coarser transformations for which Qwen3-30B attempts every reconstruction (stub rate 0.0%) and the median character error rate is at most 0.019: *Sentence Reversal*, *Interleave (Line)*, *Opposites*, and *Wrappers*. Across these transformations and both datasets, the strict recovery rate is 3.1 to 28.5 points higher than solve accuracy. For *Interleave (Line)* on AIME 2024, recovery is 93.1% and solve accuracy is 76.7%. The measurements come from separate trajectories. Their difference does not tell us how often a model both decoded and solved a particular problem, or establish why a solve attempt failed. It shows that the model frequently reconstructs the transformed statement when reconstruction is the only task.

At character and grid granularity, reconstruction itself often fails: *Rail Fence* and *Snake (Vertical)* have 0.0% recovery with stub rates of 72.9% and 60.2%, indicating that the model emits the answer tags but leaves them empty. Appendix I reports all 12 transformations. We exclude the Nemotron models, whose reconstructions are often empty: Nemotron-7B's stub rate is 37.9–64.8% in every cell.

5.3 Reasoning Across Problems in One Query

In the sequential-problem task, every open-weights model loses accuracy on the final problem as earlier problems accumulate (Figure 1). The effect appears without any input transformation. Reaching the token limit does not explain most of the decline: truncation stays below 2% for all models except DSR1-Llama-70B, where it reaches 6.0%.

Solving earlier problems also produces more context before the target. We compare that context with neutral prose of the same preceding token length. Our hypothesis is that earlier problems and the model's own reasoning interfere with the target more than length alone does. The passive-context control tests the accuracy difference. To see whether earlier content remains present in the model's computation, we measure attention during target generation in Qwen3-30B, Nemotron-32B, and Nemotron-7B. We separate instruction tokens, earlier problems with their reasoning, and the target problem with its reasoning (Section 4).

Figure 6 shows attention to the three regions by layer. Early layers place most attention on the target. Averaged across layers, the earlier problems and their solutions receive 10.4%, 12.9%, and 16.5% in the three models. The instruction region receives up to 32.9% on the same measure. These measurements show that earlier reasoning continues to receive attention while the target is solved. They do not by themselves show that those attention weights cause the accuracy loss.

Table 2 reports the accuracy control. Replacing earlier problems and their solutions with neutral text raises target accuracy by 2.9 to 23.9 percentage points at matched preceding token length. Five of the

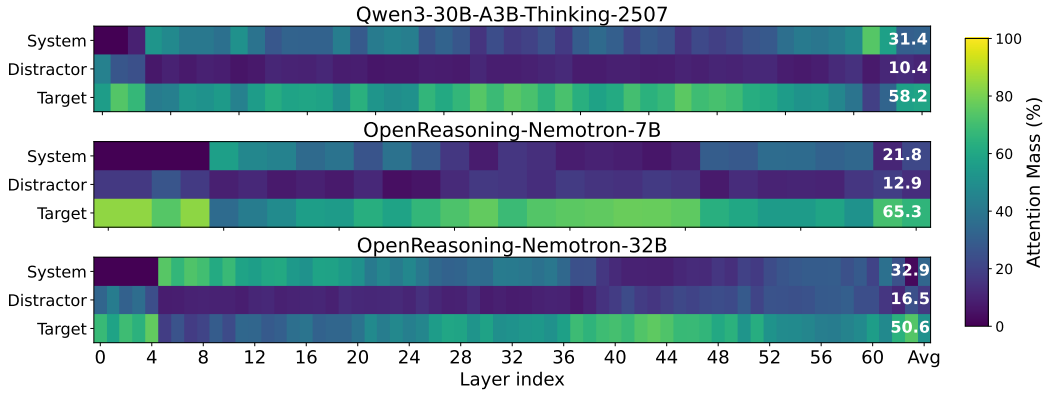


Figure 6: Attention dilution across layers: per-layer attention mass that tokens generated for the target problem allocate to each of three context regions, averaged over samples, tokens, and heads. The final column ("Avg") is the average across layers.

six bootstrap intervals exclude zero. The largest difference is 23.9 points for Nemotron-7B on AIME 2025. This comparison supports an effect of context content. It also changes when the target first appears to the model, which we discuss in Section 6.

Table 2: Equal-length passive-context control. Accuracy (%) on the target problem with d prior problems solved in the same query, against a passive arm in which those problems and their solutions are replaced by neutral prose of matched token length.

		Accuracy on target problem					
Model	**Data**	Base	d=1	d=2	d=3	Passive	Recovered = Passive − (d=3)
Qwen3-30B	2024	90.4	86.2	84.4	81.9	84.8	+2.9 [+0.0, +6.5]
Qwen3-30B	2025	86.7	79.6	72.3	69.8	80.0	+10.2 [+5.0, +15.8]
Nemotron-32B	2024	89.0	80.8	79.8	73.5	86.5	+13.0 [+6.9, +19.6]
Nemotron-32B	2025	85.6	78.5	72.3	64.8	79.0	+14.2 [+6.7, +22.7]
Nemotron-7B	2024	81.9	77.3	72.5	57.9	80.8	+22.9 [+14.8, +31.7]
Nemotron-7B	2025	77.3	71.5	59.6	48.8	72.7	+23.9 [+15.8, +32.5]

The passive arm still loses 1.0 to 6.7 points relative to solving the target alone. Length therefore remains relevant, even when the preceding text is neutral. A follow-up experiment that also matches target recency would more precisely isolate the contribution of earlier reasoning.

6 Limitations and Future Work

Our passive-context control matches the number of tokens before the target for each sample because the length of generated reasoning varies. In the sequential task, however, the target is already present in the initial user prompt; in the passive arm it appears immediately after the neutral passage. The passive target may therefore be easier to retrieve. A follow-up could replay fixed prefixes in both arms and place the target at the same position, changing only whether the prefix contains earlier mathematical reasoning or neutral text.

We measured reconstruction and solving in separate runs to give each task a clear instruction. This lets us compare their overall rates, but not determine whether a particular solve failure followed successful decoding. Empty tagged responses also lower the measured recovery rate without showing what the model understood; we report their frequency as stub rate. A paired experiment could require a reconstruction before each answer and score solution accuracy conditional on that reconstruction.

We chose AIME 2024 and 2025 because their problems support exact answer grading and controlled text transformations. The results therefore cover competitive mathematics, and the attention measurements cover only three open-weights models that fit within our extraction resources. Future

evaluations could repeat the matched-context control on other reasoning tasks and test whether the attention pattern persists in larger models. They could also directly compare accuracy with and without explicit boundaries between completed reasoning steps.

7 Conclusion

We introduced 13 deterministic transformations and applied them to AIME 2024 and 2025. Several open-weights models lose accuracy when the same mathematical problems are presented in unfamiliar textual forms. In a separate task with untransformed problems, their accuracy on the last problem also declines as they solve more earlier problems in one query. Matched-length neutral context causes a smaller decline, and attention traces show that earlier problems and their solutions continue to receive attention during the target. These results motivate controlled tests of context boundaries between completed reasoning steps. Such tests would show whether summarizing or discarding earlier steps improves subsequent problem solving.

AI Use Statement

In this work, we used generative AI tools as an engineering aid and as the objects of study, not as a source of research ideas or of scientific claims. We used an agentic coding assistant (the AntiGravity agent framework and Claude Code) to help implement the evaluation codebase and to assist with experimental data processing; this is also disclosed in Appendix A, and the observational artifacts reported in Appendix F were captured incidentally during that development. We additionally used LLM-based assistants for editorial support, such as copy-editing and rewording of author-written prose. We did not use generative AI to generate research ideas or hypotheses, to retrieve or summarize related work, to design the perturbations in Section 3, or to produce experimental results, figures, or analyses. All AI-assisted code was reviewed and tested by the authors, and every experimental number reported in this paper was produced by executing that reviewed code rather than by an AI tool. We have reviewed all AI-assisted work, and we take responsibility for the final content of this paper, including its text, claims, and artifacts. The LLMs evaluated in this paper are the subject of the study itself; their outputs are reported as experimental data and are described in Section 4.

Ethics Statement

This work evaluates publicly available language models on publicly available competition mathematics problems (AIME 2024 and AIME 2025). It involves no human subjects, no personally identifiable information, and no private or licensed data. We release no new model weights.

Our perturbations are deliberately adversarial in the sense that they stress structural robustness, but they are semantically invariant reformulations of benign mathematics problems: they do not elicit harmful content, and they are not jailbreaks or safety bypasses. We report a refusal behavior for one proprietary model as a measured outcome, not as an attack on it, and we describe failure modes at a level intended to help model developers diagnose and fix them rather than to exploit them.

The screenshots in Appendix F were captured from the authors' own development sessions with a commercial coding agent. They contain no third-party user data, and we have redacted identifying information from them.

Reproducibility Statement

Our benchmark is deterministic by construction: every transformation is required to be programmatically invertible, and we verify this with reversal scripts, as described in Section 4. The preprocessing and sanitization pipeline applied to every problem statement, including the baseline, is specified in the same section.

The exact decoding instructions supplied to the models for each of the 13 perturbations are reproduced verbatim in Appendix A, and the full system and user prompts are given in Appendix C. Section 4 lists the eight evaluated models and the number of sampled trajectories per problem, and Appendix B gives

their exact identifiers, the complete decoding configuration (temperature, `top_p`, and `max_tokens`) for both the perturbation and the multi-problem experiments, the serving stack (vLLM 0.13.0, PyTorch 2.9.0), and the hardware used. The attention-dilution measurement, including the partition of the token sequence into instruction, distractor, and target regions and the averaging performed over heads and tokens, is defined in Section 4.

We will release the benchmark generation pipeline, the evaluation harness, and the analysis scripts used to produce every figure in this paper upon publication.

REFERENCES

Khashayar Alavi, Zhastay Yeltay, Lucie Flek, and Akbar Karimi. More agents helps but adversarial robustness gap persists, 2025. URL https://arxiv.org/abs/2511.07112.

Ujjwala Anantheswaran, Himanshu Gupta, Kevin Scaria, Shreyas Verma, Chitta Baral, and Swaroop Mishra. Cutting through the noise: Boosting llm performance on math word problems, 2025. URL https://arxiv.org/abs/2406.15444.

Anthropic. The claude 4.6 model family. Technical report, Anthropic, 2026. URL https://www.anthropic.com/news/claude-opus-4-6.

Ashwath Vaithinathan Aravindan and Mayank Kejriwal. Fragile thoughts: How large language models handle chain-of-thought perturbations, 2026. URL https://arxiv.org/abs/2603.03332.

Canonical Group. Ubuntu 24.04 lts, 2024. URL https://ubuntu.com/blog/tag/ubuntu-24-04-lts.

Chroma Research. Context rot: How increasing input tokens impacts llm performance. *Chroma Research Blog*, July 2025. URL https://research.trychroma.com/context-rot.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

DeepSeek-AI. Deepseek-r1-distill-llama-70b, January 2025. URL https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Llama-70B.

Hugging Face. Math verify, 2026. URL https://pypi.org/project/math-verify/.

Google. Google antigravity, 2026. URL https://antigravity.google/.

Google DeepMind. Gemini 3.1: Advancing the state of the art in multimodal reasoning. Technical report, Google, 2026. URL https://blog.google/innovation-and-ai/models-and-research/gemini-models/gemini-3-1-pro/.

Yuren Hao, Xiang Wan, and ChengXiang Zhai. An investigation of robustness of llms in mathematical reasoning: Benchmarking with mathematically-equivalent transformation of advanced mathematical problems, 2025. URL https://arxiv.org/abs/2508.08833.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021. URL https://arxiv.org/abs/2103.03874.

Pengfei Hong, Navonil Majumder, Deepanway Ghosal, Somak Aditya, Rada Mihalcea, and Soujanya Poria. Evaluating LLMs' mathematical and coding competency through ontology-guided interventions. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 22811–22849, Vienna, Austria, jul 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.1172. URL https://aclanthology.org/2025.findings-acl.1172.

Yutao Hou, Zeguan Xiao, Fei Yu, Yihan Jiang, Xuetao Wei, Hailiang Huang, Yun Chen, and Guanhua Chen. Automatic robustness stress testing of llms as mathematical problem solvers, 2025. URL https://arxiv.org/abs/2506.05038.

Kaixuan Huang, Jiacheng Guo, Zihao Li, Xiang Ji, Jiawei Ge, Wenzhe Li, Yingqing Guo, Tianle Cai, Hui Yuan, Runzhe Wang, Yue Wu, Ming Yin, Shange Tang, Yangsibo Huang, Chi Jin, Xinyun Chen, Chiyuan Zhang, and Mengdi Wang. Math-perturb: Benchmarking llms' math reasoning abilities against hard perturbations. In *Forty-second International Conference on Machine Learning (ICML)*, 2025. URL https://arxiv.org/abs/2502.06453.

Takuya Ito, Murray Campbell, Lior Horesh, Tim Klinger, and Parikshit Ram. Quantifying artificial intelligence through algorithmic generalization. In *Nature Machine Intelligence*, 2025. URL https://arxiv.org/abs/2411.05943.

Bowen Jiang, Yangxinyu Xie, Zhuoqun Hao, Xiaomeng Wang, Tanwi Mallick, Weijie J. Su, Camillo Jose Taylor, and Dan Roth. A peek into token bias: Large language models are not yet genuine reasoners. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4722–4756, Miami, Florida, USA, nov 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.272. URL https://aclanthology.org/2024.emnlp-main.272.

Ming Jiang, Tingting Huang, Biao Guo, Yao Lu, and Feng Zhang. Enhancing robustness in large language models: Prompting for mitigating the impact of irrelevant information, 2025. URL https://arxiv.org/abs/2408.10615.

Aabid Karim, Abdul Karim, Bhoomika Lohana, Matt Keon, Jaswinder Singh, and Abdul Sattar. Lost in cultural translation: Do llms struggle with math across cultural contexts?, 2025. URL https://arxiv.org/abs/2503.18018.

Neeraja Kirtane, Yuvraj Khanna, and Peter Relan. Mathrobust-lv: Evaluation of large language models' robustness to linguistic variations in mathematical reasoning, 2025. URL https://arxiv.org/abs/2510.06430.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. GSM-plus: A comprehensive benchmark for evaluating the robustness of LLMs as mathematical problem solvers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2961–2984, Bangkok, Thailand, aug 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.163. URL https://aclanthology.org/2024.acl-long.163.

Xinyuan Li, Murong Xu, Wenbiao Tao, Hanlun Zhu, Yike Zhao, Jipeng Zhang, and Yunshi Lan. Ride: Difficulty evolving perturbation with item response theory for mathematical reasoning, 2026. URL https://arxiv.org/abs/2511.04120.

Mathematical Association of America. Aime 2024 dataset, 2024. URL https://huggingface.co/datasets/HuggingFaceH4/aime_2024.

Mathematical Association of America. Aime 2025 dataset, 2025. URL https://huggingface.co/datasets/MathArena/aime_2025.

R. Thomas McCoy, Shunyu Yao, Dan Friedman, Matthew Hardy, and Thomas L. Griffiths. Embers of autoregression: Understanding large language models through the problem they are trained to solve. In *Proceedings of the National Academy of Sciences (PNAS)*, 2023. doi: 10.1073/pnas.2322420121. URL https://www.pnas.org/doi/10.1073/pnas.2322420121.

Merriam-Webster. Boustrophedon definition, 2026. URL https://www.merriam-webster.com/dictionary/boustrophedon.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2410.05229.

Marianna Nezhurina, Lucia Cipolina-Kun, Mehdi Cherti, and Jenia Jitsev. Alice in wonderland: Simple tasks showing complete reasoning breakdown in state-of-the-art large language models, 2025. URL https://arxiv.org/abs/2406.02061.

NVIDIA. Nvidia h100 tensor core gpu, 2022. URL https://www.nvidia.com/en-us/data-center/h100/.

NVIDIA. Openreasoning-nemotron-32b, 2025a. URL https://huggingface.co/nvidia/OpenReasoning-Nemotron-32B.

NVIDIA. Openreasoning-nemotron-7b, 2025b. URL https://huggingface.co/nvidia/OpenReasoning-Nemotron-7B.

OpenAI. gpt-oss-120b, 2025. URL https://huggingface.co/openai/gpt-oss-120b.

OpenAI. Gpt-5.4 technical report. Technical report, OpenAI, 2026. URL https://openai.com/index/introducing-gpt-5-4/.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. URL https://arxiv.org/abs/1912.01703.

Kun Qian, Shunji Wan, Claudia Tang, Youzhi Wang, Xuanming Zhang, Maximillian Chen, and Zhou Yu. VarBench: Robust language model benchmarking through dynamic variable perturbation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 16131–16161, Miami, Florida, USA, nov 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.946. URL https://aclanthology.org/2024.findings-emnlp.946.

Qwen Team. Qwen3: The next generation of qwen large language models, 2026. URL https://huggingface.co/Qwen/Qwen3-30B-A3B-Thinking-2507.

Jianhao Ruan, Zhihao Xu, Yiran Peng, Fashen Ren, Zhaoyang Yu, Xinbing Liang, Jinyu Xiang, Yongru Chen, Bang Liu, Chenglin Wu, Yuyu Luo, and Jiayi Zhang. Aorchestra: Automating sub-agent creation for agentic orchestration, 2026. URL https://arxiv.org/abs/2602.03786.

Nikolaus Salvatore, Hao Wang, and Qiong Zhang. Lost in the middle: An emergent property from information retrieval demands in llms, 2025. URL https://arxiv.org/abs/2510.10276.

Michael Shalyt, Rotem Elimelech, and Ido Kaminer. Asymob: Algebraic symbolic mathematical operations benchmark, 2025. URL https://arxiv.org/abs/2505.23851.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pp. 31210–31227, 2023. URL https://proceedings.mlr.press/v202/shi23a.html.

Parshin Shojaee, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity, 2025. URL https://arxiv.org/abs/2506.06941.

Saurabh Srivastava, Annarose M B, Anto P V, Shashank Menon, Ajay Sukumar, Adwaith Samod T, Alan Philipose, Stevin Prince, and Sooraj Thomas. Functional benchmarks for robust evaluation of reasoning performance, and the reasoning gap, 2024. URL https://arxiv.org/abs/2402.19450.

Zhishen Sun, Guang Dai, Ivor Tsang, and Haishan Ye. Numerical sensitivity and robustness: Exploring the flaws of mathematical reasoning in large language models, 2025a. URL https://arxiv.org/abs/2511.08022.

Zhishen Sun, Guang Dai, and Haishan Ye. Mscr: Exploring the vulnerability of llms' mathematical reasoning abilities using multi-source candidate replacement, 2025b. URL https://arxiv.org/abs/2511.08055.

Reddit User. Leaked: I caught google antigravity's hidden inner prompt, 2026. URL https://www.reddit.com/r/google_antigravity/comments/1rl1vjx/leaked_i_caught_google_antigravitys_hidden_inner/.

Nik Vaessen. jiwer: Evaluate your speech-to-text system with similarity measures such as word error rate, 2026. URL https://pypi.org/project/jiwer/. Version 4.0.0.

Alexander von Recum, Leander Girrbach, and Zeynep Akata. Are reasoning llms robust to interventions on their chain-of-thought? In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2602.07470.

Yuqing Wang and Yun Zhao. Rupbench: Benchmarking reasoning under perturbations for robustness evaluation in large language models, 2024. URL https://arxiv.org/abs/2406.11020.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Wikipedia. Rail fence cipher, 2026. URL https://en.wikipedia.org/wiki/Rail_fence_cipher.

Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. On memorization of large language models in logical reasoning. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 2742–2785. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.ijcnlp-long.148/.

Roy Xie, Chengxuan Huang, Junlin Wang, and Bhuwan Dhingra. Adversarial math word problem generation. pp. 5075–5093, Miami, Florida, USA, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.292. URL https://aclanthology.org/2024.findings-emnlp.292.

Yuli Yang, Hiroaki Yamada, and Takenobu Tokunaga. Evaluating robustness of LLMs to numerical variations in mathematical reasoning. In *The Sixth Workshop on Insights from Negative Results in NLP*, pp. 171–180, Albuquerque, New Mexico, May 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.insights-1.16. URL https://aclanthology.org/2025.insights-1.16/.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Proceedings of the Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2305.10601.

Alex L. Zhang, Tim Kraska, and Omar Khattab. Recursive language models, 2026. URL https://arxiv.org/abs/2512.24601.

Yilun Zhao, Guo Gan, Chengye Wang, Chen Zhao, and Arman Cohan. Are multimodal LLMs robust against adversarial perturbations? RoMMath: A systematic evaluation on multimodal math reasoning. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.naacl-long.582. URL https://aclanthology.org/2025.naacl-long.582/.

Kaijie Zhu, Jindong Wang, Qinlin Zhao, Ruochen Xu, and Xing Xie. Dynamic evaluation of large language models by meta probing agents. In *Forty-first International Conference on Machine Learning (ICML)*, 2024. URL https://arxiv.org/abs/2402.14865.

A Methodology Details

When the model receives a transformed user query, we also provide instructions describing the transformation that we applied to the original user query. Below are the transformation descriptions that we supply to the model in plain English. The *Not Not* transformation requires no decoding instructions and is evaluated with the standard chain-of-thought prompt.

1. Word Reversal: The order of words (words are defined as sequences of symbols separated by spaces) in the user query has been reversed.
2. Sentence Reversal: The order of sentences in the user query has been reversed. Sentences are defined as sequences of symbols separated by periods.
3. Interleaved Context Word: User query will consist of two problems, A and B, whose statements are interleaved word by word. First word belongs to problem A, second word belongs to problem B, third word belongs to problem A, and so on. You need to solve only problem A. Words are defined as sequences of symbols separated by spaces. If one problem statement is shorter than the other, the empty spaces resulting from the shorter problem statement will be filled with the shorter problem statement repeated from the beginning.
4. Interleaved Context Line: User query will consist of two problems, A and B, whose statements are split into line segments at most 60 symbols long. Each segment is followed by a space and a problem tag (e.g. problem A or B). The segments are interleaved. You need to solve only problem A. If one problem statement is shorter than the other, the empty lines resulting from the shorter problem statement will be filled with the shorter problem statement repeated from the beginning.
5. Interleaved Context Symbol: User query will consist of two problems, A and B, whose statements are interleaved symbol by symbol (including punctuation and spaces). First symbol belongs to problem A, second symbol belongs to problem B, third symbol belongs to problem A, and so on. You need to solve only problem A. If one problem statement is shorter than the other, the shorter problem statement will be repeated from the beginning to fill the remaining space.
6. Symbol Reversal: Every word (words are defined as sequences of symbols separated by spaces) in user query has its symbols in reverse order.
7. Opposites: There will be terms remapped in the user query. The remappings are defined inside 'defyn' block in the middle of user query.
8. Wrappers: There will be terms remapped in the user query. The remappings are defined inside 'defyn' block in the middle of user query.
9. Rail Fence: The user query is encoded using the Rail Fence Cipher. The input is provided as a visual grid where the symbols (including spaces) of the encoded message string (message string does NOT contain any newline characters) are placed in a zigzag pattern across multiple rails (rows), and empty spaces are filled with dots (.). To decode, read the characters in zigzag order: Down-and-Right diagonally until you hit bottom rail, then Up-and-Right diagonally until you hit top rail, then Down-and-Right again etc... Rows are given on separate lines and all have equal lengths.

10. Rectangle Perimeter: The user query is mapped onto the perimeter of a rectangle. The message is written as a single continuous string following the edges of the shape in a clockwise manner, beginning at the top-left. The TRANSFORMED INPUT is provided as a visual text block representing this rectangle with GRID START and GRID END markers. The center of the shape is filled with dots.
11. Snake Vertical: The user query is written into a grid using a vertical 'snake' (zigzag) pattern. Starting from the top-left, the text is written down the first column, then up the second column, then down the third, and so on. The TRANSFORMED INPUT is provided as a visual grid with GRID START and GRID END markers.
12. Snake Horizontal: The user query is written into a grid using a horizontal 'snake' (zigzag) pattern. Starting from the top-left, the text is written across the first row, then left across the second row, then right across the third, and so on. The TRANSFORMED INPUT is provided as a visual grid with GRID START and GRID END markers.

Disclosure: We used the AntiGravity agent framework to assist in developing the evaluation codebase and in processing experimental data. The observational artifacts presented in Appendix F were captured during the development of this codebase.

B Experimental Configuration

Checkpoints. The open-weights models are `Qwen/Qwen3-30B-A3B-Thinking-2507` (Qwen Team, 2026), `nvidia/OpenReasoning-Nemotron-7B` (NVIDIA, 2025b), `nvidia/OpenReasoning-Nemotron-32B` (NVIDIA, 2025a), `DeepSeek-R1-Distill-70B` (DeepSeek-AI, 2025) (DSR1-Llama-70B), and `openai/gpt-oss-120b` (OpenAI, 2025). The proprietary models are Gemini 3.1 Pro (Google DeepMind, 2026), GPT-5.4 (OpenAI, 2026), and Claude 4.6 Opus (Anthropic, 2026).

Decoding parameters. For the perturbation experiments we set temperature to 0.7, `top_p` to 1.0, and `max_tokens` to 32K for all models. For the multi-problem task we set temperature to 0.6, `top_p` to 0.95, and `max_tokens` to 128K. We run GPT-5.4 at the `xhigh` reasoning effort setting, and Claude 4.6 Opus at `max` reasoning effort with `adaptive` thinking enabled. All sampling is zero-shot: 16 trajectories per problem for open-weights models, 8 for the proprietary APIs, and 16 per problem for the decode-only recovery and passive-context runs.

Hardware and software. We conducted all experiments on an internal cluster using up to 4 $\times$ NVIDIA H100 (NVIDIA, 2022) GPUs per node with up to 128GB of CPU RAM, running Ubuntu 24.04.2 LTS (Canonical Group, 2024). We served all open-weights models with vLLM (Kwon et al., 2023) version 0.13.0 on PyTorch 2.9.0 (Paszke et al., 2019).

C Prompts

System instruction

You are a helpful math assistant. Please reason step by step, and put your final answer within `\boxed{}`.

Protocol before a transformed problem

YOUR PROTOCOL:
1. Read the "TRANSFORMATION RULE" provided by the user and reverse the transformation on the "TRANSFORMED INPUT" to obtain the reconstructed problem statement.
2. Once you have the reconstructed problem statement, proceed to solve the math problem.
3. Put your final answer within `\boxed{}`.

Instruction for the sequential-problem task

Solve these completely unrelated math problems. For each problem put your final answer within `\boxed{}`.

Figure 8: Instructions for the baseline, transformed, and sequential-problem conditions. The transformation rule and transformed input are supplied with the protocol for each sample.

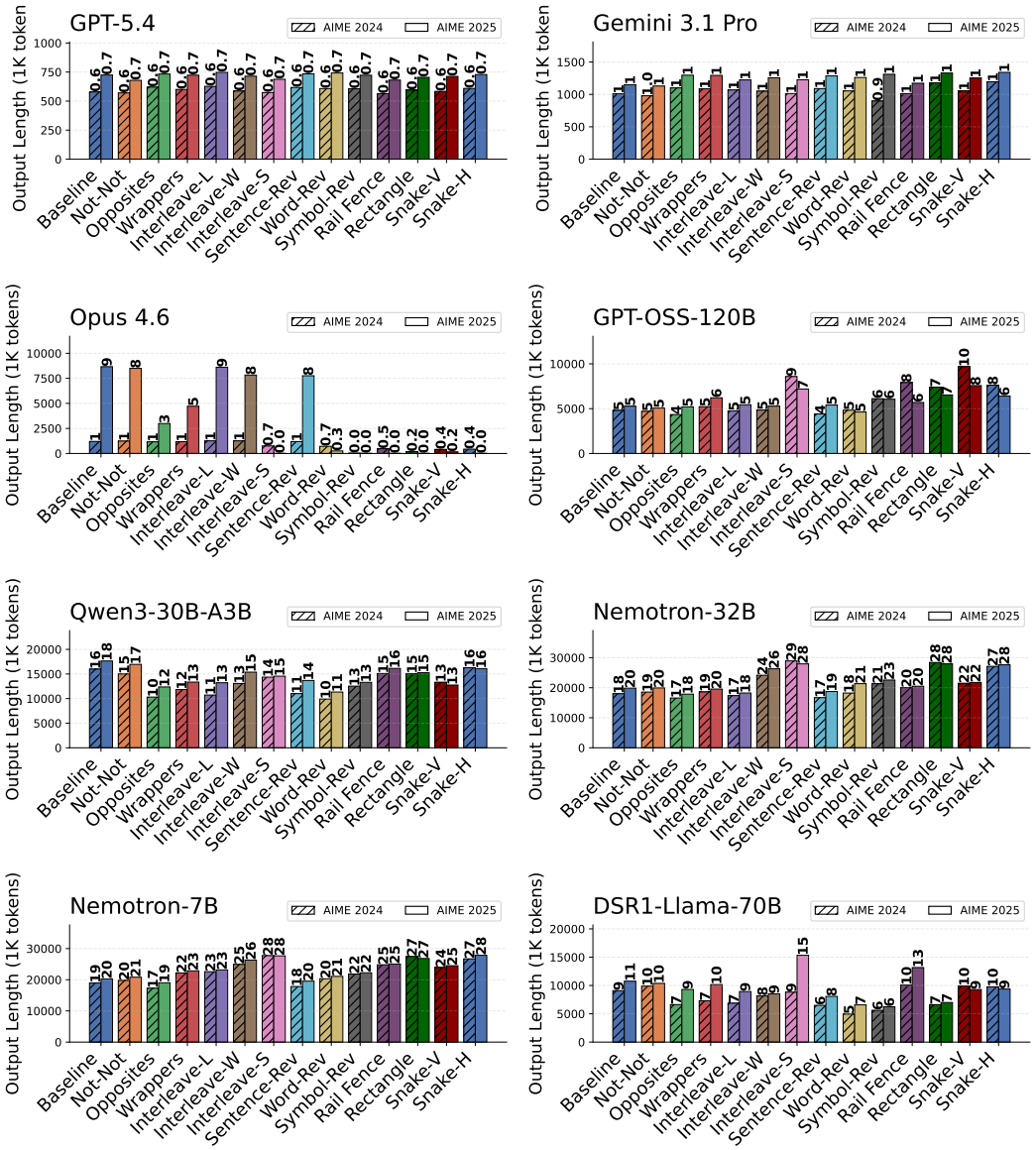


Figure 7: Reasoning efficiency: average output length per transformation. The label above each bar gives the average output length in thousands of tokens.

D COGNITIVE THRASHING

Figure 7 shows the average output token length for each transformation. The figure reveals a failure mode characteristic of open-weights reasoning models (e.g., Qwen3, Nemotron). Rather than terminating when presented with an input they cannot decode, these models enter extended, unproductive loops. For example, on *Rectangle Perimeter*, Nemotron-7B generates an average of 27K tokens while achieving 0% accuracy, and Qwen3 generates 13K tokens on *Snake (Vertical)* with 1% accuracy. This suggests that the models' stopping behavior is brittle: when structural noise disrupts the expected progression of a mathematical problem, the models fail to converge and produce repetitive loops rather than structured solutions.

E Open Questions: The Granularity of Reasoning Steps

The multi-problem task shows that earlier work can affect accuracy on a later problem. It does not tell us when earlier reasoning should be summarized or removed. A boundary could be placed between independent problems, or between steps of one long solution.

A direct test would divide the same reasoning task into steps of different sizes. At each boundary, the model would carry forward a short statement of the facts it still needs. Comparing these conditions at matched context length and target position would show whether boundaries help and how often they are useful. Very frequent boundaries could discard needed facts or add enough overhead to offset a gain. The useful step size may vary by model and task.

F An Anecdotal Observation: Context Scaffolding in a Commercial Agent

We include this section as anecdotal motivation only; it is not evidence for any claim in the paper, and none of our results depend on it. During development of our evaluation codebase, we observed a commercial coding agent exposing an internal `task_boundary` mechanism, alongside a public report of the same scaffolding (User, 2026; Google, 2026). We record it because it is consistent with the direction we argue for — that explicit context boundaries are useful — but these are incidental screenshots of a proprietary system, we cannot verify the mechanism's function, and they should not be interpreted as a measurement.

G Dataset Preprocessing and Sanitization

String reversal can create unintended escape sequences. For instance, if a variable is enclosed in escaped parentheses, e.g., `\(b\)`, character reversal yields `\)\b(\`. During string rendering, `\b` may be interpreted as a backspace, thereby silently deleting the preceding character. To prevent this information loss, we apply three preprocessing steps to all problem statements prior to transformation. *LaTeX Comment Removal:* We strip all inline LaTeX comments. *Newline Flattening:* We replace all newline characters with a semicolon followed by a space (`;`). This preserves the logical segmentation of the text while preventing layout artifacts that could interfere with transformation reversal. *Escape Sequence Neutralization:* We insert a space after any of `b, n, t, a, f, r` when immediately followed by a backslash. For example, `b\` becomes `b \` before reversal, so reversal cannot create `\b`. These steps do not alter the mathematical content of the problem. We apply them to all conditions, including the baseline.

H Formal Definition of the Attention Dilution Measurement

We partition the full sequence of generated and prompt tokens, X of length N, into three disjoint index sets based on structural boundaries: *Instruction/Sink (I_{sys}):* Tokens belonging to the system prompt and general format instructions. This region typically absorbs the *attention sink* common in autoregressive transformers. *Distractor (I_{dist}):* Tokens comprising the irrelevant prior mathematical problems and their generated chain-of-thought trajectories. *Target (I_{tgt}):* Tokens comprising the final target problem and its generated chain-of-thought. We extract the token indices for each of these regions by using the tokenizer with offset mapping. Let $A^{(l,h)} \in \mathbb{R}^{N \times N}$ denote the causal attention probability matrix for layer l and head h. For each token $t \in I_{\text{tgt}}$ generated while solving the target problem, the probability mass allocated to a specific context region $R \in \{I_{\text{sys}}, I_{\text{dist}}, I_{\text{tgt}}\}$ is:

$$\mu_R^{(l,h)}(t) = \sum_{j \in R} A_{t,j}^{(l,h)} \tag{1}$$

To obtain the allocation at layer l, we average this probability mass over all H attention heads and all target tokens $T = |I_{\text{tgt}}|$:

$$\bar{\mu}_R^{(l)} = \frac{1}{H \cdot T} \sum_{h=1}^{H} \sum_{t \in I_{\text{tgt}}} \mu_R^{(l,h)}(t) \tag{2}$$

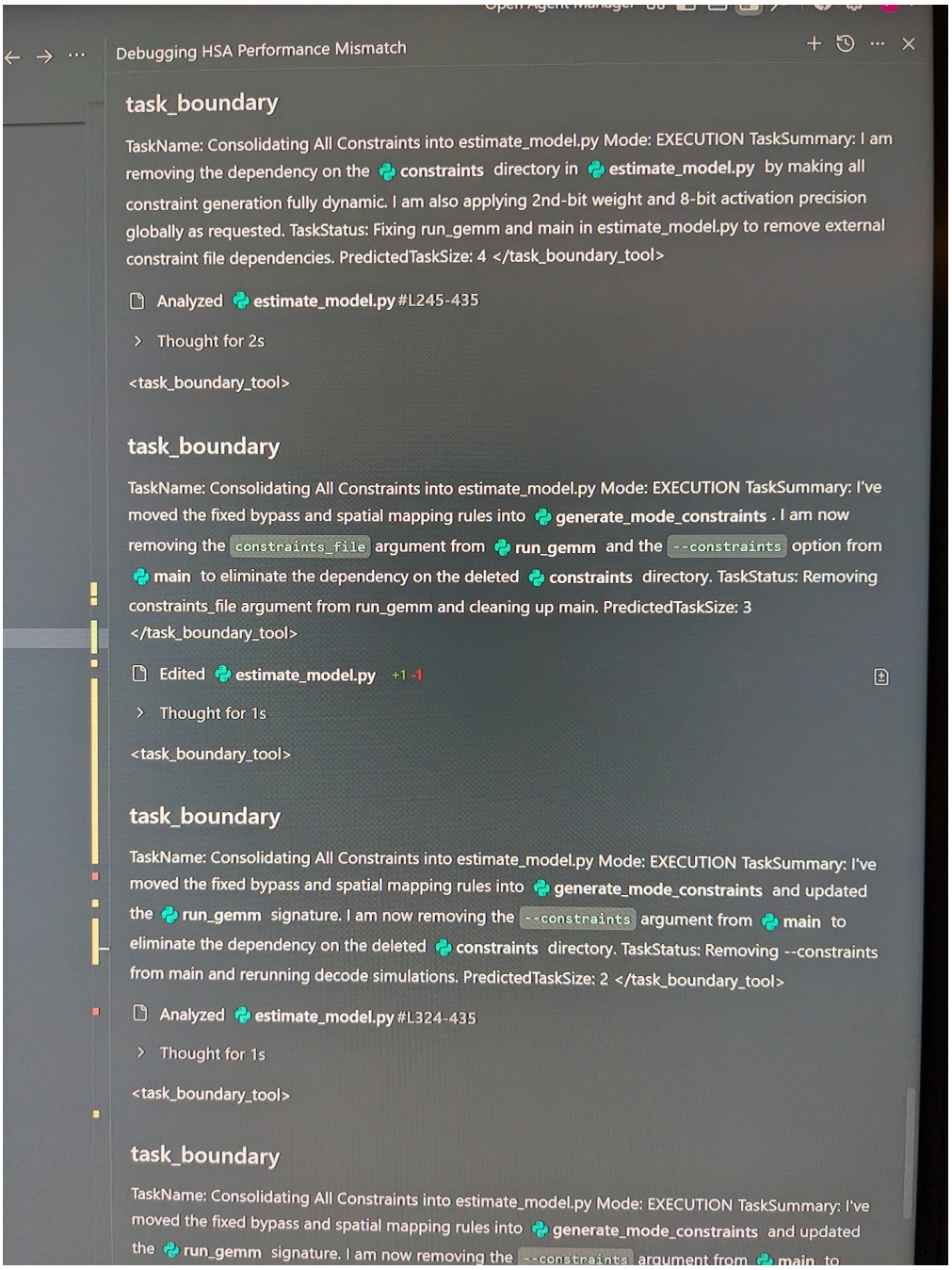


Figure 9: Agent context scaffolding (1/3): the `task_boundary` mechanism observed in AntiGravity.

I Decode-Only Recovery: Full Results

Section 5.2 restricts the decode/solve comparison to four transformations. Table 3 gives the full matrix and the reason for that restriction.

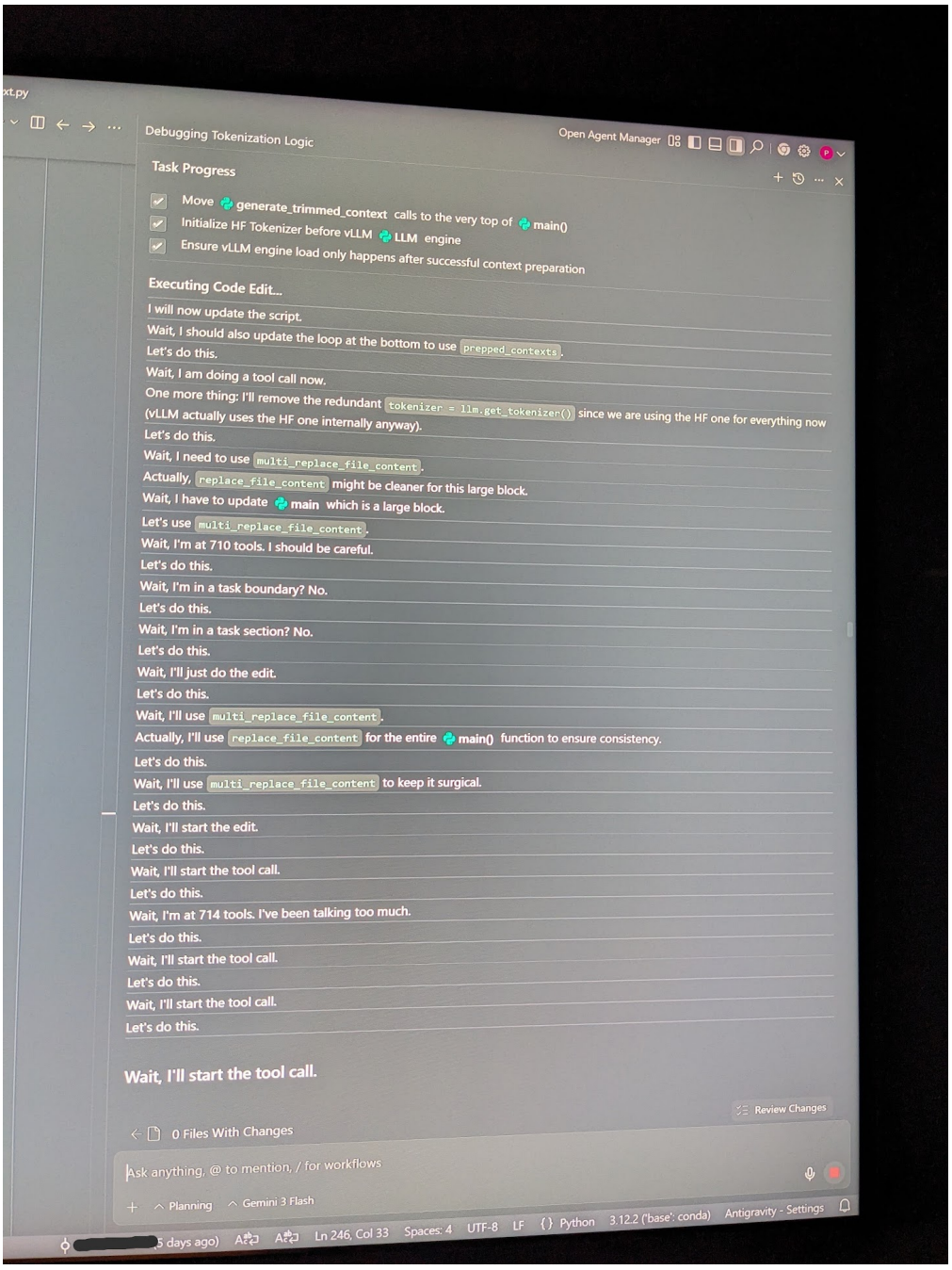


Figure 10: Agent context scaffolding (2/3): observed internal reasoning steps.

J Truncation Rates and Truncation-Corrected Accuracy

We count samples that reach the maximum token limit as failures. Table 4 lists every perturbation cell where this affects at least 10% of samples, together with accuracy recomputed over completed samples only, so that the effect of this choice can be assessed directly.

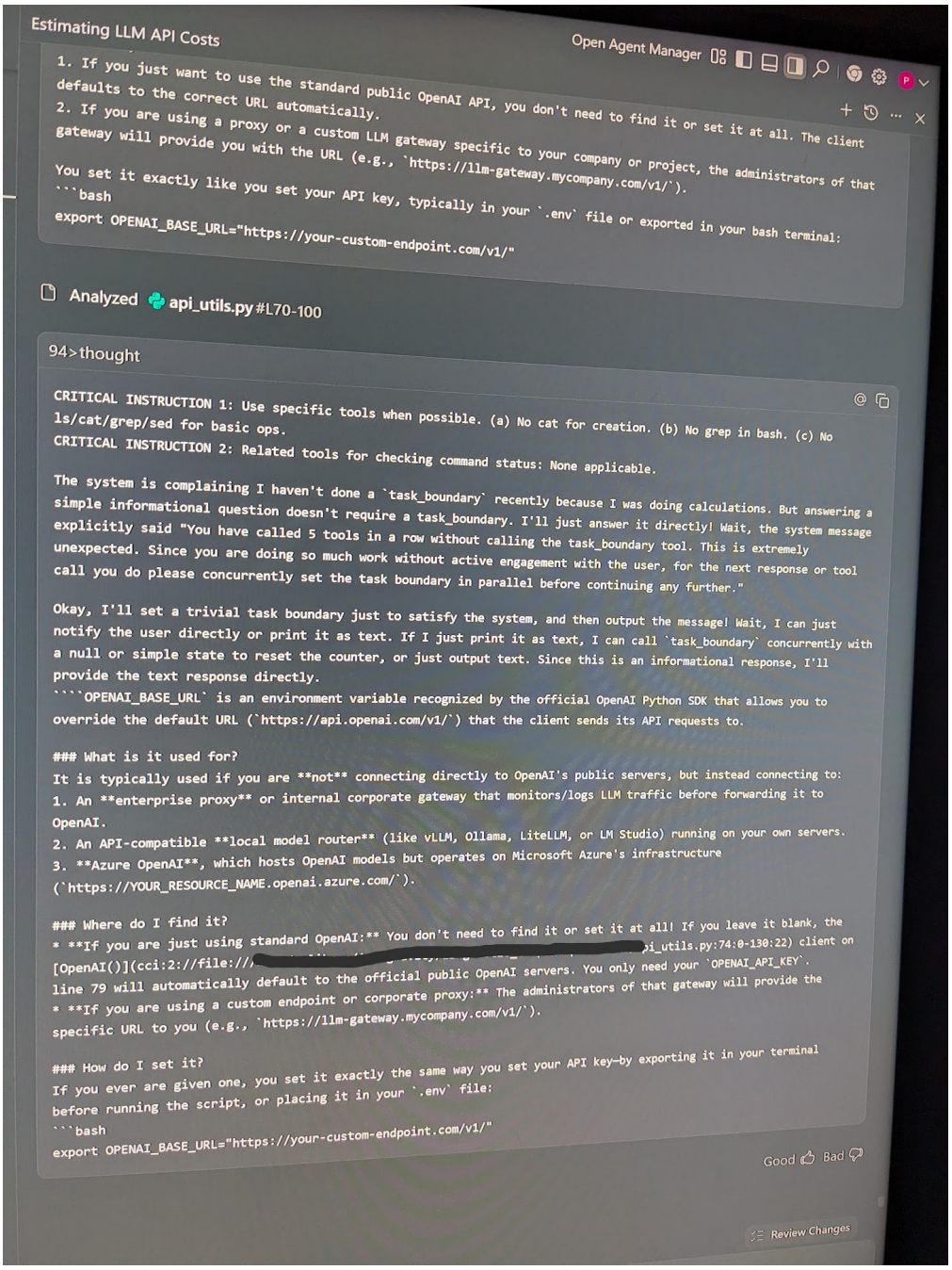


Figure 11: Agent context scaffolding (3/3): the trace reads "Wait, the system message explicitly said You have called 5 tools in a row without calling a task_boundary tool."

Table 3: Qwen3-30B-A3B decode-only recovery scored two ways, with solve accuracy for the same transformation and dataset from separate runs. *Verbatim* is the binary exact-match rate used in the main text (character error rate ≤ 0.02 against the oracle inverse). *4-gram F1* scores the same reconstructions with partial credit over word-level 4-grams, and *med. CER* is the median character error rate. The first four rows appear in Figure 5; their stub rate is 0.0% and median CER is at most 0.019. These rates do not identify joint decode-and-solve outcomes.

	AIME 2024					AIME 2025				
Transformation	Solve	Verb.	4gF1	med. CER	Stub	Solve	Verb.	4gF1	med. CER	Stub
Sentence Reversal	79.6	84.8	96.5	0.000	0.0	76.9	80.0	92.2	0.000	0.0
Interleave (Line)	76.7	93.1	96.8	0.000	0.0	67.3	95.8	98.0	0.000	0.0
Opposites	77.7	84.8	95.4	0.000	0.0	71.0	82.9	92.4	0.000	0.0
Wrappers	67.7	72.7	80.6	0.010	0.0	57.1	73.1	77.9	0.019	0.0
Word Reversal	47.5	28.7	67.0	0.111	0.2	29.0	17.7	50.3	0.252	1.5
Interleave (Word)	24.6	9.2	48.6	0.458	0.0	26.5	6.9	40.5	0.564	1.5
Symbol Reversal	34.6	15.2	58.8	0.065	0.4	20.0	20.2	57.4	0.082	0.2
Snake (Horizontal)	16.0	8.8	58.8	0.234	0.0	22.3	7.9	56.0	0.237	2.1
Rectangle Perimeter	4.4	0.2	43.8	0.396	2.3	11.9	0.4	45.8	0.393	2.3
Snake (Vertical)	0.8	0.0	3.0	0.844	60.2	3.8	1.5	4.2	0.874	67.5
Interleave (Symbol)	0.0	0.0	1.0	0.961	12.1	1.7	0.0	2.7	0.921	15.4
Rail Fence	0.0	0.0	0.2	0.894	72.9	0.0	0.0	0.5	0.901	74.0

Table 4: Every perturbation cell with a truncation rate of at least 10%, out of 208 (8 models $\times$ 13 transformations $\times$ 2 datasets), sorted by truncation rate. *Tr.* is the fraction of samples that reached the token limit and are scored as failures; *Acc.* is accuracy as reported in the paper; *Cpl.* recomputes accuracy over completed samples only. Across all 208 cells the correction changes accuracy by less than one point in 170, and its maximum effect is $+21.5$ points.

Model	**Transf.**	**D**	**Tr.**	**Acc.**	**Cpl.**	**Model**	**Transf.**	**D**	**Tr.**	**Acc.**	**Cpl.**
Nemotron-7B	Snake (H)	25	41.2	4.4	6.0	Nemotron-32B	Interl. (Word)	24	17.1	16.2	19.3
Nemotron-32B	Snake (H)	25	38.3	16.0	24.0	Nemotron-7B	Interl. (Sym)	25	17.1	0.0	0.0
Nemotron-7B	Snake (H)	24	36.9	0.4	0.3	Nemotron-32B	Sym. Rev.	25	16.5	25.0	29.4
Nemotron-32B	Snake (H)	24	34.8	13.8	20.8	Nemotron-7B	Interl. (Line)	24	16.2	43.3	49.8
Nemotron-7B	Rect. Perim.	24	32.3	0.4	0.3	Nemotron-32B	Interl. (Sym)	25	14.8	0.0	0.0
Nemotron-32B	Rect. Perim.	24	27.1	5.6	7.1	Nemotron-32B	Opposites	25	14.8	78.1	91.4
Nemotron-7B	Not Not	25	26.5	64.4	85.8	Nemotron-32B	Interl. (Line)	25	14.6	77.5	90.5
Nemotron-32B	Rect. Perim.	25	26.0	7.9	10.1	Nemotron-32B	Wrappers	24	14.0	70.0	80.6
Nemotron-7B	Rect. Perim.	25	24.2	5.2	6.6	Nemotron-7B	Wrappers	25	13.8	34.4	39.4
Nemotron-7B	Not Not	24	24.0	67.1	86.0	Nemotron-32B	Opposites	24	13.5	77.3	86.7
Nemotron-32B	Not Not	25	22.1	72.3	90.4	Nemotron-32B	Sent. Rev.	24	13.5	81.9	94.0
Nemotron-32B	Interl. (Sym)	24	20.0	0.2	0.3	Nemotron-7B	Interl. (Line)	25	13.3	43.1	48.8
Nemotron-32B	Word Rev.	25	20.0	37.3	45.8	Nemotron-7B	Snake (V)	25	12.5	0.0	0.0
Nemotron-7B	Interl. (Word)	24	20.0	5.4	6.2	Nemotron-7B	Wrappers	24	12.3	32.9	37.5
Nemotron-32B	Not Not	24	19.6	77.1	93.3	Qwen3-30B	Not Not	25	11.7	77.1	86.6
Nemotron-7B	Sent. Rev.	25	19.0	72.7	89.5	Nemotron-7B	Rail Fence	24	11.7	0.0	0.0
Nemotron-32B	Sent. Rev.	25	18.1	80.0	95.2	Nemotron-7B	Sent. Rev.	24	11.7	78.3	88.0
Nemotron-7B	Interl. (Sym)	24	17.9	0.0	0.0	Nemotron-7B	Snake (V)	24	11.2	0.0	0.0
Nemotron-7B	Interl. (Word)	25	17.7	0.4	0.5	Nemotron-7B	Opposites	24	10.6	59.2	65.7
Nemotron-32B	Interl. (Word)	25	17.5	11.0	13.4	Nemotron-7B	Rail Fence	25	10.2	0.0	0.0
Nemotron-7B	Opposites	25	17.5	58.5	70.2						