Do LLMs Overthink Basic Math Reasoning? Benchmarking the Accuracy-Efficiency Tradeoff in Language Models

Anonymous ACL submission

Abstract

Large language models (LLMs) achieve impressive performance on complex mathematical benchmarks yet sometimes fail on basic math reasoning while generating unnecessarily verbose responses. In this paper, we present a **systematic benchmark and comprehensive empirical study** to evaluate the efficiency of reasoning in LLMs, focusing on the fundamental tradeoff between accuracy and overthinking. **First,** we formalize the *accuracy-verbosity tradeoff*. **Second,** we introduce the *Overthinking Score*, a harmonic-mean metric combining accuracy and token-efficiency for holistic model evaluation. **Third,** we establish an evaluation protocol with dynamically-generated data across **14** basic math tasks. **Fourth,** we conduct a large-scale empirical study evaluating **53** LLMs, including reasoning and quantized variants across different reasoning budgets. Our findings reveal: ***1)*** model performance on complex benchmarks does not translate directly to basic math reasoning; ***2)*** reasoning models generate $\sim$**18**$\times$ **more tokens** while sometimes achieving **lower accuracy** and exhibit catastrophic collapse when token is constrained, dropping by $\sim$**28%**; ***3)*** the accuracy-verbosity relationship is non-monotonic with extended reasoning budgets yielding diminishing returns (GPT-5/o-series models show zero accuracy gain from **low** $\rightarrow$ **medium** $\rightarrow$ **high** reasoning effort). *Our findings challenge the assumption that longer reasoning in LLMs necessarily improves mathematical reasoning.*

1 Introduction

Modern large language models (LLMs) (Abdin et al., 2025; Yang et al., 2025a; Xu et al., 2025) often produce long chain-of-thought when answering arithmetic and other deterministic tasks. Intuitively, more reasoning tokens should help models avoid mistakes by giving them room to compute. However, sometimes models often generate verbose traces that do not improve final correctness and, under constrained budgets, can degrade accuracy (Lin et al., 2025; Qu et al., 2025). *This reveals a fundamental issue:* while models can produce sophisticated-looking reasoning chains, they often generate excessive verbosity that neither improves accuracy nor demonstrates genuine understanding. This phenomenon is know as *overthinking in LLMs*.

Consider a model computing $234 + 567$. While humans solve this with minimal steps, language models often generate hundreds of tokens explaining place values, carrying operations, and mathematical principles; yet sometimes arrive at incorrect answers. Recent studies (Srivastava et al., 2025; Yan et al., 2025) show this paradox: models achieving 90% accuracy on GSM8K may score below 40% on basic addition. More concerning, reasoning models specifically trained for deeper thinking (Abdin et al., 2025; Shojaee*† et al., 2025) perform *worse* while generating more tokens. This inverse relationship between verbosity and accuracy suggests that current models conflate explanation with understanding (Zhang et al., 2024), producing text that superficially resembles reasoning without actual problem-solving capability.

Previous work has approached this problem from different angles. Chain-of-thought prompting (Wei et al., 2022) and its variants (Yao et al., 2023a; Besta et al., 2024; Pandey et al., 2025; Yao et al., 2023b; Jin et al., 2024; Ranaldi et al., 2023; Li et al., 2025a) encourage step-by-step reasoning but increase computational costs without guaranteeing accuracy improvements (Huang et al., 2025; Chen et al., 2025a). Recent studies on overthinking (Chen et al., 2025b; Sui et al., 2025) focus on mitigation strategies like early stopping (Pu et al., 2025) or self-breaking (Zhao et al., 2025), but lack principled metrics to quantify the phenomenon. Efficiency-focused evaluations (Li et al., 2025c) measure thinking time or API calls but treat accuracy and efficiency as independent dimensions, missing their fundamental tradeoff. Most critically,

existing benchmarks evaluate only final accuracy, remaining blind to computational waste that makes models impractical for deployment.

Three key gaps remain in proper understanding of overthinking: ***(1)*** absence of principled metrics that jointly measure correctness and efficiency, as existing benchmarks treat these them separate dimensions rather than a tradeoff; ***(2)*** reliance on static benchmarks vulnerable to contamination; and ***(3)*** lack of robust parsing *(fair evaluation)* to extract answers from diverse model outputs reliably. These gaps prevent distinguishing genuine reasoning from performative verbosity. To address these gaps: ***1)*** we introduce the **Overthinking Score**, a harmonic-mean metric that penalizes imbalance between accuracy and token efficiency, forcing models to be both correct and concise. ***2)*** we design a dynamic test generation protocol across 14 basic math reasoning tasks ensuring fresh, reproducible evaluation. **Third**, we conduct large-scale empirical studies studying efficiency-accuracy tradeoffs missed by accuracy-only metrics.

Our systematic benchmark evaluation of 53 LLMs across multiple experimental conditions yields striking insights: ***1)*** The Overthinking Score reveals efficiency-accuracy tradeoffs missed by accuracy-only metrics, e.g., smaller models outperforming larger ones through better token efficiency. ***2)*** Reasoning models produce $\sim$**6,780** tokens on average versus 378 for standard models while performing worse; accuracy collapses under token limits (from 72% to 44% at 1,024 tokens); and quantized models retain strong reasoning ability, implying overthinking arises from training rather than hardware constraints. ***3)*** Extended reasoning budgets demonstrate sharply diminishing returns, with models showing minimal accuracy gains beyond moderate token allocations while introducing contradictions and error accumulation in long chains, revealing fundamental limitations in current reasoning paradigms. ***4)*** Through comprehensive ablation studies including quantization effects, constrained generation, and reasoning budget analysis, our benchmark provides a standardized framework for evaluating reasoning efficiency that complements existing accuracy-focused benchmarks.

2 Related Work

Mathematical Reasoning in LLMs LLMs have achieved impressive performance on benchmark datasets such as GSM8K (Cobbe et al., 2021), GSM-Plus (Li et al., 2024), MATH (Hendrycks et al., 2021), and HARD-Math (Fan et al., 2024). Yet recent work has highlighted a striking gap: models that excel at word problems often fail on simple arithmetic (Srivastava et al., 2025; Yan et al., 2025; Xu and Ma, 2024). To expose these weaknesses, new benchmarks of simple numerical tasks have been proposed (Li et al., 2025b; Rahman, 2025), showing that models struggle with core numeracy skills despite advanced reasoning capabilities elsewhere. Several approaches aim to improve mathematical reasoning. For instance, Graph of Thoughts (Besta et al., 2024) structures intermediate reasoning as a graph, improving performance on algorithmic tasks such as sorting. LogicPuzzleRL (Wong et al., 2025) uses reinforcement learning on custom logic puzzles to strengthen algebraic, geometric, and combinatorial reasoning. While these approaches enhance problem-solving on complex tasks, they provide limited insight into why models underperform on simple arithmetic or how to measure reasoning efficiency.

Overthinking in LLMs A complementary line of research shows that LLMs often produce unnecessarily long reasoning chains for trivial problems (Chen et al., 2025b). This phenomenon, termed *overthinking*, wastes computation and sometimes reduces accuracy. Mitigation strategies include efficiency-oriented prompting, reasoning-output pruning, and model-based interventions (Sui et al., 2025). Specific methods such as ThoughtTerminator (Pu et al., 2025), Self-Braking Tuning (Zhao et al., 2025), and process supervision approaches (Lightman et al., 2023; Luo et al., 2024) actively shorten reasoning or improve step-level verification without hurting accuracy. However, prior work largely emphasizes mitigation heuristics rather than providing a principled way to *quantify* overthinking. Without robust metrics, it remains unclear how to diagnose or compare models' efficiency in reasoning. Our work introduces the **Overthinking Score** as a principled joint metric, and validating it through dynamic task generation and large-scale experiments.

3 Problem Formulation

3.1 Task Space Definition

We formalize the evaluation of basic mathematical reasoning through a task space $\mathcal{T}$ =

$\{\tau_1, \tau_2, ..., \tau_K\}$ comprising $K = 14$ deterministic arithmetic operations. Each task τ_i represents a mapping from an input domain $\mathcal{X}_i$ to an output domain $\mathcal{Y}_i$, where the ground truth function $f_i : \mathcal{X}_i \to \mathcal{Y}_i$ is computationally deterministic and mathematically well-defined.

Definition 1 (Deterministic Arithmetic Task)**.** *A deterministic arithmetic task $\tau = (\mathcal{X}, \mathcal{Y}, f, \mathcal{C})$ consists of:*

- *An input space $\mathcal{X} \subseteq \mathbb{Z}^n$ or $\mathcal{X} \subseteq \mathbb{Z} \times \mathbb{Z}$*
- *An output space $\mathcal{Y} \subseteq \mathbb{Z}$ or $\mathcal{Y} \subseteq \mathbb{Q}$ or $\mathcal{Y} \subseteq \mathcal{P}(\mathbb{Z})$*
- *A deterministic function $f : \mathcal{X} \to \mathcal{Y}$ with complexity $\mathcal{C} \in \{O(n), O(n \log n), O(n^2)\}$*

Table 1 presents our complete task suite spanning four categories. For list-based operations, inputs are defined as $\mathcal{L} = \{l_1, l_2, ..., l_n\}$ where $l_i \in \mathbb{Z}$ and $n \in \mathbb{N}$. Complete mathematical foundations for each task, including uniqueness proofs and complexity analysis, are provided in Appendix C. Experimental configuration details including sampling strategies, hyperparameters, and computational resources are described in Appendix E.

Cat.	Task	Input	Output
Basic	Sorting	$\mathcal{L} \subset \mathbb{Z}^n$	Ordered list
	Comparison	(a, b)	$\{>, <, =\}$
	Sum	$\mathcal{L} \subset \mathbb{Z}^n$	$\sum_i \mathcal{L}_i$
	Multiplication	$\mathcal{L} \subset \mathbb{Z}^n$	$\prod_i \mathcal{L}_i$
	Division	$(a, b), b \neq 0$	a/b
	Abs. Diff	(a, b)	$\lvert a - b \rvert$
Ext.	Find Max	$\mathcal{L} \subset \mathbb{Z}^n$	$\max(\mathcal{L})$
	Find Min	$\mathcal{L} \subset \mathbb{Z}^n$	$\min(\mathcal{L})$
Stats	Mean	$\mathcal{L} \subset \mathbb{Z}^n$	$\frac{1}{n} \sum_i \mathcal{L}_i$
	Median	$\mathcal{L} \subset \mathbb{Z}^n$	$\text{med}(\mathcal{L})$
	Mode	$\mathcal{L} \subset \mathbb{Z}^n$	$\arg\max_v f(v)$
Count	Odd Count	$\mathcal{L} \subset \mathbb{Z}^n$	$\lvert\{x : x \bmod 2 = 1\}\rvert$
	Even Count	$\mathcal{L} \subset \mathbb{Z}^n$	$\lvert\{x : x \bmod 2 = 0\}\rvert$

Table 1: Task suite comprising 14 deterministic tasks with formally specified input-output mappings.

3.2 The Accuracy-Verbosity Space

We conceptualize model behavior in a two-dimensional space $\mathcal{S} = [0, 1] \times \mathbb{R}^+$ representing accuracy and average token count. Given a language model M with parameters θ, the response generation process is:

$$r = M(p(x, \tau_i); \theta, \phi) \tag{1}$$

where $p : \mathcal{X}_i \times \mathcal{T} \to \mathcal{S}$ is a prompt construction function and $\phi = \{T, p_{top}, \text{max_tokens}\}$ represents inference hyperparameters.

Definition 2 (Overthinking)**.** *A model M exhibits **overthinking** on task τ_i if there exists a more concise model M' such that:*

$$\mathbb{E}_{x \sim \mathcal{D}_i}[\mathbb{I}[\mathcal{P}(M'(x)) = f_i(x)]] \geq \tag{2}$$

$$\mathbb{E}_{x \sim \mathcal{D}_i}[\mathbb{I}[\mathcal{P}(M(x)) = f_i(x)]] \tag{3}$$

$$\mathbb{E}_{x \sim \mathcal{D}_i}[|M'(x)|] < \mathbb{E}_{x \sim \mathcal{D}_i}[|M(x)|] \tag{4}$$

3.3 The Overthinking Score

We introduce a principled metric for quantifying the accuracy-verbosity tradeoff. For a model M evaluated on task τ_i, let A_i denote its accuracy (proportion of correct answers) and $\bar{T}_i$ denote the average number of output tokens generated per instance. To normalize token counts across models, we define:

Definition 3 (Token Efficiency)**.** *Let T_{min} and T_{max} denote the minimum and maximum average token counts observed across all models on all tasks in our evaluation. The token efficiency for model M on task τ_i is:*

$$E_{t,i} = 1 - \frac{\bar{T}_i - T_{min}}{T_{max} - T_{min}} \tag{5}$$

where $E_{t,i} \in [0, 1]$, with higher values indicating more efficient (concise) generation.

Definition 4 (Overthinking Score)**.** *The Overthinking Score for model M on task τ_i combines accuracy and token efficiency via harmonic mean:*

$$\mathcal{O}_i = \frac{2 \cdot A_i \cdot E_{t,i}}{A_i + E_{t,i}} \tag{6}$$

The harmonic mean formulation ensures critical properties including boundedness ($\mathcal{O}_i \in [0, 1]$), symmetry, and severe penalization of imbalance (see Appendix A for complete proofs and sensitivity analysis). A model achieving 90% accuracy with 10% efficiency scores only $\mathcal{O} = 0.18$, while a balanced model with 60% in both dimensions achieves $\mathcal{O} = 0.60$. We complement this metric with supplementary measurements including average token count, word count, and character count for comprehensive evaluation of the accuracy-efficiency balance.

3.4 Alternative Formulations

We evaluated several aggregation functions before adopting the harmonic mean. The arithmetic mean ($\mathcal{O}_{arith} = \frac{A+E_t}{2}$) fails to penalize imbalance adequately-a model with perfect accuracy but 10% efficiency still scores 0.55. The geometric mean provides better balance sensitivity but lacks the strong penalty that characterizes overthinking behavior. Among all symmetric, homogeneous means, the harmonic mean maximally penalizes imbalance while maintaining smooth differentiability (detailed comparison and proofs in Appendix B).

4 Experimental Methodology

4.1 Dynamic Test Generation

To ensure evaluation free from data contamination, we implement a dynamic test generation protocol. Let $\mathcal{G} : \Theta \times \mathcal{S} \to \mathcal{D}$ represent our generation function where $\Theta = \{N, F, [r_{min}, r_{max}], \mathcal{L}, s\}$ specifies generation parameters. For each task τ and fold $f \in \{1, ..., F\}$, we generate test instances by: ***1)*** Setting seed $s_f = \text{hash}(s, f, \tau)$ for reproducibility; ***2)*** Sampling list lengths from $\mathcal{L} = \{8, 16, 32, 64\}$ for list-based tasks (except multiplication which uses $\{2, 4, 8\}$ numbers to avoid overflow); ***3)*** Drawing values from $\text{Uniform}[-1000, 1000]$ ensuring numerical diversity; ***4)*** Computing ground truth $y = f_\tau(x)$ for validation

We use $N = 1000$ samples per fold with $F = 3$ folds for open-source models (100 samples for closed-source due to cost), generating 42,000 unique problems per model. The complete generation algorithm, validation, and reproducibility protocols are provided in Appendix F and Appendix K.

4.2 Model Selection and Inference

We evaluate 53 models spanning multiple families (GPT (Achiam et al., 2023), Gemini (Comanici et al., 2025), Llama (Grattafiori et al., 2024), Mistral (Parmar et al., 2024), Qwen (Yang et al., 2025b), Phi (Abdin et al., 2025)) with parameters from 0.5B-72B. This includes base, instruction-tuned, and reasoning variants to isolate the impact of training objectives on overthinking behavior.

4.3 Prompt Construction and Response Parsing

For each test instance, we construct standardized prompts: $p(x_i, \tau) = \text{desc}(\tau) \oplus \text{format}(x_i) \oplus \text{instr}$ where instr = "Your final answer must be in the format \boxed{answer}". Complete prompt templates for all 14 tasks are provided in Appendix I.

Extracting answers from diverse model outputs requires robust parsing. Through analysis of 5000+ responses, we developed a hierarchical extraction system achieving 98.7% success rate (detailed design principles and validation procedures in Appendix J): ***1) Primary:*** Extract content within \boxed{} patterns; ***2) Secondary:*** Parse explicit answer markers ("The answer is..."); ***3) Tertiary:*** Extract from code blocks or markdown formatting ***4) Fallback:*** Apply task-specific heuristics.

Task-specific validation ensures extracted answers represent solutions rather than input echoes. We employ a hierarchical extraction strategy that prioritizes LaTeX boxed notation, then explicit answer markers, code block outputs, and finally last-line heuristics. The complete end-to-end pipeline (Algorithm 4) and hierarchical answer extraction algorithm (Algorithm 5) are detailed in Appendix F.

4.4 Statistical Analysis

We use k-fold cross-validation with $k = 3$, reporting mean and standard deviation across folds. Token counting uses model-specific tokenizers when available, with api response for closed-source models. Statistical significance testing, error taxonomy, and detailed empirical analysis including correlation studies are presented in Appendix G.

5 Results and Insights

We evaluated 53 models across basic mathematical tasks, quantization settings, and constrained generation budgets (refer to Table 2 for raw numbers). Our high level results show that *larger models or models that produces more thinking tokens are not reliably better*. Models that perform well on complex benchmarks often fail at elementary operations; models trained to reason more tend to produce long, confident-looking text that sometimes *hurts* correctness; and quantization often preserves basic arithmetic ability for large models.

5.1 The Basic-Math Paradox

Finding 1 — Basic-Math Paradox

Models with high complex benchmark performance degrade on basic math reasoning. Performance is task-specific and suggests reliance on memorized heuristics rather than compositional computation.

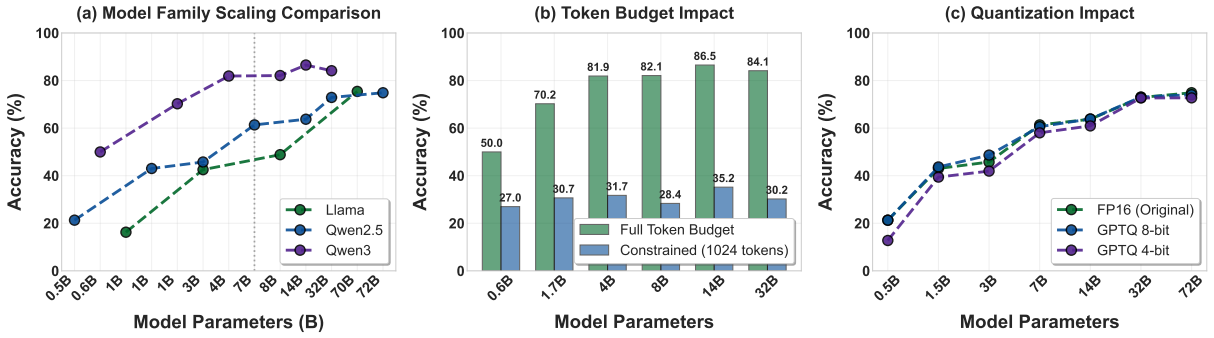

Figure 1: Three dimensions of our evaluation. **(a) Scaling performance across different families:** *Llama* shows large jumps between small and large models. *Qwen2.5* shows generally monotonic but sublinear scaling. *Qwen3* exhibits non-monotonic behavior with 14B outperforming 32B. **(b) Token budget constraints:** *Qwen3 reasoning models* accuracy under full budget vs. *1024-token limit* reveals catastrophic degradation. **(c) Quantization robustness:** *Qwen2.5 family* across FP16, 8-bit, and 4-bit precision shows size-dependent tolerance to compression.

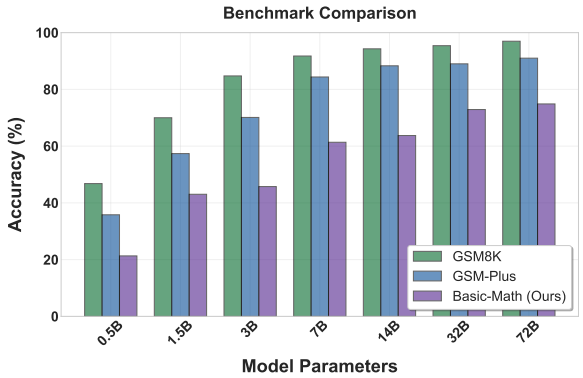

Figure 2: **Benchmark paradox:** *Qwen2.5 family* performance on GSM8K, GSM-Plus, and basic math reasoning shows significant discrepancies across models.

Figure 2 shows the discrepancy between benchmark performance (GSM8K, GSM-Plus) and basic arithmetic capability for the Qwen2.5 family. Models achieving over 95% on GSM8K score below 75% on our basic math tasks, revealing that complex benchmark performance does not transfer to fundamental operations. This pattern suggests models learn *heuristics* or pattern-matching that work for the training distribution of complex benchmarks rather than compositional arithmetic ones.

5.2 Scaling Effects: Non-Monotonic and Diminishing Returns

Finding 2 — Non-Monotonic Scaling

Parameter scaling shows diminishing and sometimes negative returns. Larger models do not guarantee better performance, with mid-sized variants often achieving superior efficiency-accuracy balance.

Figure 1(a) shows scaling behavior across three model families, revealing that parameter count does not guarantee better basic math performance. Within the *Qwen3* family we observe **non-monotonic scaling**: **Qwen3-14B = 86.52%** vs **32B = 84.13%**-a decline of **2.39%** despite 2.3× more parameters. By contrast, smaller scale steps yield large gains: **Qwen3-0.6B = 49.99%** → **1.7B = 70.24%** (+**20.25** points, +**40.51**% relative). The *Qwen2.5* family shows monotonic but sublinear improvement: **Qwen2.5-14B = 63.74%** → **32B = 72.90%** (+**9.16** points), indicating that scaling helps in some regimes but with diminishing returns. The *Llama* family shows large jumps between small and large models with significant variance across parameter counts. The vertical dotted line at 7B marks where performance curves begin to flatten, showing diminishing returns from further scaling. As models scale they can accumulate competing heuristics: some improve benchmark-style problems while others interfere on low-level operations, producing the non-monotonic scaling we observe. Notably, **GPT-4.1-mini = 90.23%** slightly edges **GPT-4.1 = 89.88%**, showing smaller/optimized variants can be practically preferable.

5.3 Overthinking: Verbosity Hurts Correctness

Finding 3 — Overthinking

Reasoning models produce long, repetitive explanations that sometimes reduce accuracy. Models tend to produce reasoning-like text rather than performing correct computation.

Models	Parameters (B)	Accuracy (Avg%)	Instruction Following (Avg%)	Overthinking Score (Avg)	Output Tokens (Avg)	Output Words (Avg)	Output Chars (Avg)
		Mean ± *Std*	*Mean* ± *Std*				
Qwen3 Family							
Qwen3	0.6	49.99 ± 3.53	83.85 ± 2.23	0.545	3162.8	1620.9	8301.8
Qwen3	1.7	70.24 ± 3.62	86.54 ± 2.35	0.647	3157.2	1620.7	8445.9
Qwen3	4	81.90 ± 3.00	91.57 ± 2.00	0.698	3091.2	1623.1	8489.9
Qwen3	8	82.10 ± 3.17	91.58 ± 2.13	0.704	3027.8	1584.6	8260.3
Qwen3	14	**86.52** ± 2.94	99.27 ± 2.29	0.727	3607.6	1941.2	10556.1
Qwen3	32	84.13 ± 2.80	93.05 ± 3.13	0.668	2845.9	1497.5	7790.1
Phi Family							
Phi-4	14	**78.92** ± 3.27	97.46 ± 1.06	0.863	378.6	194.6	989.9
Phi-4-mini (I)	3.8	54.55 ± 3.80	95.02 ± 1.23	0.697	292.1	146.6	684.9
Phi-4-reasoning-plus	14	69.54 ± 3.50	88.89 ± 1.80	0.234	6780.7	3972	23893
Phi-4-reasoning	14	**72.23** ± 3.30	96.21 ± 1.50	0.352	6066.2	3710.8	23866.8
Phi-4-mini-reasoning	3.8	70.16 ± 3.40	89.56 ± 1.78	0.646	3171.9	1571.7	8450.5
Phi-3-mini-128k (I)	3.8	35.82 ± 3.74	96.58 ± 0.79	0.566	89.4	40.6	208.9
Phi-3-medium-4k (I)	14	43.47 ± 4.05	89.87 ± 2.27	0.543	189.3	109.6	553.6
Phi-3-medium-128k (I)	14	40.76 ± 4.10	96.26 ± 1.27	0.568	140	74.8	367.3
Llama Family							
Llama-3.2 (I)	1	16.25 ± 2.17	47.15 ± 3.97	0.278	336.3	159	756.9
Llama-3.2 (I)	3	42.54 ± 3.49	89.88 ± 1.73	0.591	279.7	144.6	694.7
Llama-3.1 (I)	8	48.84 ± 3.63	85.66 ± 2.53	0.646	366.4	203.4	977.7
Llama-3.1 (I)	70	**75.43** ± 3.49	98.12 ± 4.06	0.848	251.2	135.5	654.6
Llama-3.3 (I)	70	**74.59** ± 3.44	97.40 ± 5.80	0.840	312.8	174.1	859.7
Qwen2.5 Family							
Qwen2.5 (I)	0.5	21.31 ± 2.35	77.57 ± 2.75	0.348	432.3	223.2	1144.5
Qwen2.5 (I)	1.5	43.03 ± 3.48	85.45 ± 2.26	0.596	264.7	134.1	626.7
Qwen2.5 (I)	3	45.75 ± 3.50	92.35 ± 1.09	0.619	331.3	176.5	861.4
Qwen2.5 (I)	7	61.36 ± 3.99	96.47 ± 0.99	0.750	286.9	149.5	747.2
Qwen2.5 (I)	14	63.74 ± 3.82	97.83 ± 0.63	0.769	260.2	137.1	685.7
Qwen2.5 (I)	32	72.90 ± 3.77	99.26 ± 2.69	0.832	260.9	139.1	673.6
Qwen2.5 (I)	72	**74.87** ± 3.60	97.12 ± 5.00	0.840	339.2	179.8	887.4
Other Open-Source Models							
SmolLM2 (I)	1.7	16.69 ± 2.40	68.98 ± 3.19	0.285	213	93.5	481.5
Mistral-7B (I)	7	27.66 ± 3.09	96.26 ± 0.72	0.431	207.1	113.7	585.9
Mistral-Nemo (I)	12	35.43 ± 3.40	82.95 ± 2.94	0.517	377	234.2	1123.7
Qwen2.5-Math (I)	1.5	51.43 ± 3.97	94.04 ± 1.61	0.656	397.1	210	1076.9
Qwen2.5-Math (I)	7	**60.68** ± 4.01	94.36 ± 1.75	0.740	411.7	221.5	1156
OpenAI Models							
GPT-4.1	–	89.88 ± 2.50	97.79 ± 1.80	0.927	338.8	152.2	759.5
GPT-4.1-mini	–	**90.23** ± 2.30	98.14 ± 1.50	0.930	328.8	145.1	740.7
GPT-4.1-nano	–	75.35 ± 3.20	95.58 ± 2.10	0.843	338.8	148.7	760.3
GPT-4o	–	**87.56** ± 2.40	99.42 ± 1.20	0.918	290.5	154.8	749
GPT-4o-mini	–	75.00 ± 2.80	97.67 ± 1.90	0.841	341.3	172	848.7
Google Models							
Gemini-2.0-flash-lite	–	**73.33** ± 2.90	99.58 ± 1.10	0.836	215.5	116.9	523
Gemini-2.0-flash	–	**69.60** ± 3.10	94.44 ± 2.00	0.811	234.5	118.9	517.9
Gemini-2.5-flash	–	55.18 ± 3.80	63.49 ± 2.70	0.705	186.3	104.8	539.5

Table 2: Comprehensive performance evaluation of 53 language models on basic mathematical reasoning tasks. Within each model family, **bold** indicates best accuracy and underline indicates second-best accuracy. All metrics represent averages across 14 tasks with 1,000 samples each (open-source) or 100 samples (closed-source). Instruct-tuned models are marked with (I). For complete quantization and constrained evaluation results, see Appendix M.

Task	Model	Correct?	Tokens	Token Ratio	Reasoning Pattern
Sum (8 nums)	Phi-4	✓	272	1×	Direct step-by-step
	Phi-4-reasoning	✓	4,837	17.8×	Redundant verification, wasteful
Complex Sort	Phi-4	✓	89	1×	Direct output
	Phi-4-reasoning	✓	312	3.5×	Helpful decomposition (justified)
Division	Phi-4	✓	47	1×	Minimal steps
	Phi-4-reasoning	✓	1,456	31×	Triple verification, no benefit
Multiplication	Phi-4	✓	156	1×	Sequential multiply
	Phi-4-reasoning	✗	3,214	20.6×	Over-analyzed, arithmetic error
Mean calc	Phi-4	✓	198	1×	Sum then divide
	Phi-4-reasoning	✗	2,891	14.6×	Lost in verification, wrong final

Table 3: **Correct and incorrect reasoning outcomes.** Green : Efficient and correct. Yellow : Helpful structured reasoning where verbosity aids correctness. Red : Wasteful overthinking: either no benefit despite high token cost, or incorrect answers due to over-verification.

As shown in Table 2, models trained or fine-tuned to produce long chain-of-thoughts (COT) often generate more tokens yet obtain *lower* accuracy than concise variants: for example, **Phi-4** scores **78.92%** with $\sim$ 378.6 tokens on average, while **Phi-4-reasoning** scores **72.23%** with $\sim$ 6066.2 tokens: an accuracy gap of **6.69** points ($\approx$**-8.48%** relative) despite using $\sim$ 16.02$\times$ more tokens. When constrained to 1,024 tokens, Phi-4-reasoning falls to **53.48%** (an absolute loss of **18.75** points, $\approx$**-25.96%** relative from 72.23%), and Phi-4-reasoning-plus drops from **69.54%** to **44.33%** (an absolute **-25.21** points, $\approx$**-36.25%** relative). Even under constraint, Phi-4-reasoning uses $\sim$ 1013.5 tokens on average, which is about **2.68**$\times$ the token usage of concise Phi-4. This pattern suggests COT supervision shifts the optimization target toward producing plausible intermediate text rather than verifiable intermediate computations, i.e., the model learns to generate *plausible-looking exploration* that increases likelihood but is not reliably grounded in correct arithmetic, which explains why penalizing verbosity or enforcing step-level checks often recovers accuracy. To understand *which reasoning patterns help versus hurt*, Table 3 compares Phi-4 and Phi-4-reasoning on identical tasks, while Figure 3 presents actual model outputs (detailed case studies in Appendix H).

(a) Extreme Overthinking

Task: $846 \div 18$
Phi-4 (47 tok):

```
846/18=47 \boxed{47}
```

Phi-4-reas (1,456 tok):

```
Long div...
V1: 47x18=846
V2: factorization
  846=2x3x3x47
  18=2x3x3
V3: estimation
  846~900, 18~20
  900/20=45
\boxed{47}
```

Result: 31x, no gain

(b) Pathological Failure

Task: Mode([5,5,5,7,7,9])
Expected: 5
Qwen3-4B (1,024 tok):

```
Frequency:
5: 3 times
7: 2 times
9: 1 time

Mode=5nnnnnnnnnnn
nnnnnnnnnnnnnnnnn
nnnnnnnnnnnnnnnnn
nnnnnnnnnnnnnnnnn
nnnnnnnnnnnnnnnnn
[... 900+ 'n' ...]
```

Result: Stopping failure

(c) Helpful Reasoning

Task: Sort 8 numbers
Phi-4-reas (312 tok):

```
1: Separate
  Neg:-789,-456,-123
  Pos:847,456,234,
      901,123
2: Sort negs
  -789<-456<-123
3: Sort pos
  123<234<456<
  847<901
4: Combine
\boxed{[-789,-456,
-123,123,234,...]}
```

Result: 3.5x justified

(d) Self-Contradiction

Task: Max([12,45,23,89,34])
O3-Mini (2,400 tok):

```
Scan: max=89

Wait, reconsider...
is 89 largest?
Maybe 45... no 89>45

Verify:
89>12? Yes
89>45? Yes...
Duplicates? None.
Recalculate...
[1000+ tokens]
\boxed{89}
```

Result: Self-doubt loop

Figure 3: **Reasoning Pattern Case Studies: (a)** Extreme overthinking with 31x token waste through 3x verification. **(b)** Pathological failure showing stopping mechanism breakdown (∞ character repetition). **(c)** Helpful reasoning where 3.5x verbosity provides verifiable steps. **(d)** Self-contradiction loops. Complete case study in Appendix H.

5.4 Performance Under Token Constraints

Finding 4 — Efficiency Cliff

Constraining tokens causes sudden accuracy collapses, showing models lack adaptive stopping and rely on long, fixed-length exploration.

When we restrict output length, non-reasoning models remain relatively stable while reasoning models suffer sharp, sometimes catastrophic drops. Figure 1(b) visualizes this phenomenon for the Qwen3 family, showing dramatic accuracy degradation when generation is constrained to 1,024 tokens. Their accuracy follows a steep curve: below certain token budgets (e.g., $\sim$512 tokens) performance collapses; between 512–1024 tokens results are unstable; above large budgets they improve but still lag concise models. For example, **Phi-4** achieves $\mathbf{78.92}\%$ with ~ 378.6 tokens on average, whereas **Phi-4-reasoning** scores $\mathbf{72.23}\%$ while using ~ 6066.2 tokens ($\approx\mathbf{16}\times$ more); constraining Phi-4-reasoning to a 1,024-token budget reduces its accuracy to $\mathbf{53.48}\%$ (an $\mathbf{18.75}$ point absolute loss, $\approx\mathbf{26}\%$ relative), and Phi-4-reasoning-plus shows a similar, even larger collapse when constrained (see Appendix M.2 for complete results). Allowing the long-form reasoning models thousands of extra tokens recovers some accuracy but at a very small marginal return (roughly $\mathbf{3.7}\%$ per additional 1,000 tokens in the Phi example) and often still leaves these models behind concise baselines; reasoning models appear to rely on a fixed, large exploration budget rather than adapting computation to problem difficulty.

5.5 Thinking Budget: Diminishing Returns and Error Accumulation

Finding 5 — Thinking Budget

Extra thinking token budget gives small returns and increases the risk of contradictory or noisy reasoning.

Allocating larger reasoning budgets produces minimal accuracy improvements while introducing systematic inefficiencies. Figure 4 illustrates this across three model families. For Gemini models, increasing budget from *disabled* to *24,576* tokens yields only **2% absolute gain** (90% $\rightarrow$ 92%) for Gemini-2.5-Pro, while token consumption grows substantially. More strikingly, GPT-5 achieves identical **97% accuracy** on both medium and high effort levels despite using progressively more tokens, showing **zero** marginal benefit from extended reasoning. The O-series models (O3, O3-mini, O4-mini) exhibit similar plateaus: O3 maintains **97% accuracy** across all effort levels (low, medium, high) while generating $\sim$8% more tokens at higher budgets. This pattern reveals models lack *adaptive computation*: they cannot modulate reasoning depth based on problem difficulty. Instead, they apply fixed exploration patterns regardless of whether a task requires simple arithmetic or multi-step verification, resulting in systematic waste on easy problems and insufficient adaptation on hard ones. Empirically, the accuracy curve follows $A(b) = A_{\max}(1 - e^{-b/\tau}) + \epsilon(b)$, where $\tau \approx 1000$ tokens captures saturation scale and $\epsilon(b)$ represents error accumulation. Beyond moderate budgets (1,000–3,000 tokens), models often degrade by contradicting earlier reasoning,

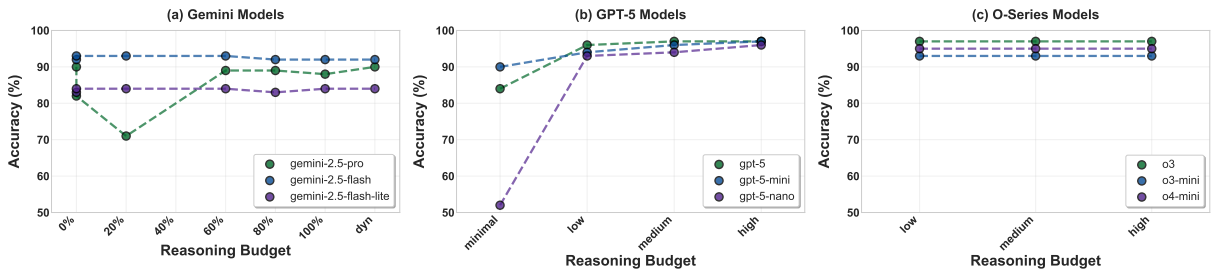

Figure 4: **Reasoning budget analysis across Gemini, GPT-5, and O-series models.** Increasing the reasoning budget yields minimal gains, showing diminishing returns and near-plateau performance at higher effort levels.

overwriting correct steps with plausible but incorrect ones; extended generation sometimes amplifies such errors (see Table 9 in Appendix L).

5.6 Quantization Robustness

Finding 6 — Quantization Robustness

Even aggressive quantization has minimal effect on Large models whereas in case of small it hurts the performance.

Quantization shows a clear size-dependent pattern illustrated in Figure 1(c): larger models tolerate aggressive compression with minimal accuracy loss, while very small models suffer large relative drops; for instance, the Qwen2.5-32 model moves from $\mathbf{73.08}\%$ (8-bit) to $\mathbf{72.67}\%$ (4-bit), a negligible change, whereas Qwen2.5-0.5 falls from $\mathbf{21.29}\%$ to $\mathbf{12.77}\%$ under 4-bit quantization ($\approx\mathbf{40}\%$ relative loss). Mid-sized models show modest penalties (e.g., $\sim 3\% - 5\%$ relative in several 14B examples), which implies that redundancy in large models preserves low-precision arithmetic behavior and that the primary issues we observe (overthinking, token brittleness) are behavioral and not simply precision-limited. Complete quantization results are presented in Appendix M.1.

5.7 Insights from the Overthinking Score Metric

Finding 7 — Overthinking Score Insights

When efficiency is included, mid-sized instruction-tuned models often become the most practical choices.

As shown in Table 2, when we combine accuracy and efficiency into the Overthinking Score, model rankings change: some smaller or mid-sized instruction-tuned models beat larger, verbose ones. The score highlights a sweet-spot around mid-large sizes (roughly 14–20B) where models have enough capacity for correct computation without developing overthinking habits. For example, **GPT-4.1-mini** scores $\mathbf{0.768}$ vs **GPT-4.1** at $\mathbf{0.752}$ (an absolute $\mathbf{0.016}$ improvement, $\approx\mathbf{2.1}\%$ relative), while **Qwen3-14B** (score $\mathbf{0.725}$) outperforms Qwen3-32B ($\mathbf{0.627}$) by $\mathbf{0.098}$ points ($\approx\mathbf{13.5}\%$ relative); conversely, reasoning-tuned variants suffer large penalties on this metric (for example, Phi-4-reasoning at $\mathbf{0.352}$ vs Phi-4 at $\mathbf{0.593}$, a $\sim 40\%$ relative deficit), showing that efficiency-aware evaluation identifies different, deployment-relevant winners than accuracy-only comparisons. The inverted-U with size suggests two things: scale gives models the ability to store and execute correct computational patterns, but excessive scale (without appropriate inductive bias) enables learned behaviors that favor verbose, high-likelihood text. Instruction tuning often helps move models toward the left side of that curve (better efficiency) if the tuning emphasizes concise, correct answers.

6 Conclusion

Our evaluation of 53 LLMs reveals: ***1)*** Parameter scaling gives diminishing and non-monotonic returns: mid-sized models (14-20B) often match or outperform larger siblings on accuracy-efficiency tradeoffs. ***2)*** "Thinking" supervision often fails: models trained for long chains typically generate $18\times$ more tokens while losing accuracy and collapsing under token constraints. ***3)*** Extended reasoning budgets show diminishing returns and error accumulation, revealing models lack adaptive stopping and step-level verification.

Practical Recommendations **Models:** GPT-4.1-mini (O-Score: 0.768), Qwen3-14B (0.725), Qwen2.5-7B for budget deployments. **Prompting:** Using formats (\boxed{}), applying token budgets strategically, avoiding open-ended "think step-by-step" on basic tasks. **Future work:** Models need adaptive stopping, step-level verification, and fixes for pathological fails (∞ repetition).

Limitations

Our study has several limitations that warrant discussion. **First,** we focus exclusively on basic mathematical operations, which may not fully capture the complexity of reasoning required in real-world problem-solving scenarios. While these fundamental tasks reveal critical insights about model capabilities, they represent only one dimension of mathematical understanding. **Second,** our automated evaluation framework, while enabling large-scale assessment across 50+ models, lacks the nuanced interpretability that human evaluation could provide. Future work could complement our quantitative metrics with qualitative analysis of reasoning patterns. **Third,** although our dynamic test generation protocol prevents direct memorization, models may still leverage statistical patterns learned during training. The extent to which models genuinely compute versus pattern-match remains an open question. **Fourth,** our evaluation depends on task-specific answer parsers that must adapt as model output formats evolve, potentially introducing systematic biases in extraction accuracy.

Ethics Statement

This work presents a systematic benchmark for assessing mathematical reasoning and overthinking tradeoffs in language models and does not generate harmful content or introduce direct ethical risks. Our evaluation uses publicly available models on deterministic mathematical tasks, requiring no collection of personal data or human annotations that could raise privacy concerns. All models are evaluated in accordance with their respective licensing agreements and usage policies.

AI Assistance: We used ChatGPT assistance for parts of the Appendix, such as generating LaTeX code for tables and refining text written by the authors. All AI-generated content was carefully reviewed and revised by the authors to ensure accuracy and clarity.

References

Marah Abdin, Sahaj Agarwal, Ahmed Awadallah, Vidhisha Balachandran, Harkirat Behl, Lingjiao Chen, Gustavo de Rosa, Suriya Gunasekar, Mojan Javaheripi, Neel Joshi, Piero Kauffmann, Yash Lara, Caio César Teodoro Mendes, Arindam Mitra, Besmira Nushi, Dimitris Papailiopoulos, Olli Saarikivi, Shital Shah, Vaishnavi Shrivastava, and 4 others. 2025. Phi-4-reasoning technical report. *Preprint*, arXiv:2504.21318.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. 2025a. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025b. Do not think that much for 2+3=? on the overthinking of o1-like llms. *Preprint*, arXiv:2412.21187.

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. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Jingxuan Fan, Sarah Martinson, Erik Y. Wang, Kaylie Hausknecht, Jonah Brenner, Danxian Liu, Nianli Peng, Corey Wang, and Michael P. Brenner. 2024. Hardmath: A benchmark dataset for challenging problems in applied mathematics. *Preprint*, arXiv:2410.09988.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

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

Kerui Huang, Shuhan Liu, Xing Hu, Tongtong Xu, Lingfeng Bao, and Xin Xia. 2025. Reasoning efficiently through adaptive chain-of-thought compression: A self-optimizing framework. *arXiv preprint arXiv:2509.14093*.

Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Zheng Li, Ruirui Li, Xianfeng Tang, Suhang Wang, Yu Meng, and 1 others. 2024. Graph chain-of-thought: Augmenting large language models by reasoning on graphs. *arXiv preprint arXiv:2404.07103*.

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

Feiyang Li, Peng Fang, Zhan Shi, Arijit Khan, Fang Wang, Dan Feng, Weihao Wang, Xin Zhang, and Yongjian Cui. 2025a. Cot-rag: Integrating chain of thought and retrieval-augmented generation to enhance reasoning in large language models. *arXiv preprint arXiv:2504.13534*.

Haoyang Li, Xuejia Chen, Zhanchao XU, Darian Li, Nicole Hu, Fei Teng, Yiming Li, Luyu Qiu, Chen Jason Zhang, Qing Li, and Lei Chen. 2025b. Exposing numeracy gaps: A benchmark to evaluate fundamental numerical abilities in large language models. *Preprint*, arXiv:2502.11075.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. 2024. Gsm-plus: A comprehensive benchmark for evaluating the robustness of llms as mathematical problem solvers. *Preprint*, arXiv:2402.19255.

Zhiyuan Li, Yi Chang, and Yuan Wu. 2025c. Thinkbench: Evaluating thinking efficiency and chain-of-thought quality of large reasoning models. *Preprint*, arXiv:2505.22113.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *Preprint*, arXiv:2305.20050.

Junhong Lin, Xinyue Zeng, Jie Zhu, Song Wang, Julian Shun, Jun Wu, and Dawei Zhou. 2025. Plan and budget: Effective and efficient test-time scaling on large language model reasoning. *arXiv preprint arXiv:2505.16122*.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. 2024. Improve mathematical reasoning in language models by automated process supervision. *Preprint*, arXiv:2406.06592.

Tushar Pandey, Ara Ghukasyan, Oktay Goktas, and Santosh Kumar Radha. 2025. Adaptive graph of thoughts: Test-time adaptive reasoning unifying chain, tree, and graph structures. *arXiv preprint arXiv:2502.05078*.

Jupinder Parmar, Shrimai Prabhumoye, Joseph Jennings, Mostofa Patwary, Sandeep Subramanian, Dan Su, Chen Zhu, Deepak Narayanan, Aastha Jhunjhunwala, Ayush Dattagupta, and 1 others. 2024. Nemotron-4 15b technical report. *arXiv preprint arXiv:2402.16819*.

Xiao Pu, Michael Saxon, Wenyue Hua, and William Yang Wang. 2025. Thoughtterminator: Benchmarking, calibrating, and mitigating overthinking in reasoning models. *Preprint*, arXiv:2504.13367.

Xiaoye Qu, Yafu Li, Zhaochen Su, Weigao Sun, Jianhao Yan, Dongrui Liu, Ganqu Cui, Daizong Liu, Shuxian Liang, Junxian He, and 1 others. 2025. A survey of efficient reasoning for large reasoning models: Language, multimodality, and beyond. *arXiv preprint arXiv:2503.21614*.

Roussel Rahman. 2025. Large language models in numberland: A quick test of their numerical reasoning abilities. *Preprint*, arXiv:2504.00226.

Leonardo Ranaldi, Giulia Pucci, Federico Ranaldi, Elena Sofia Ruzzetti, and Fabio Massimo Zanzotto. 2023. Empowering multi-step reasoning across languages via tree-of-thoughts. *arXiv preprint arXiv:2311.08097*.

Parshin Shojaee*†, Iman Mirzadeh*, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. 2025. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity.

Gaurav Srivastava, Shuxiang Cao, and Xuan Wang. 2025. Towards reasoning ability of small language models. *Preprint*, arXiv:2502.11569.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, and Xia Hu. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *Preprint*, arXiv:2503.16419.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le

Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zhen Hao Wong, Jingwen Deng, Runming He, Zirong Chen, Qijie You, Hejun Dong, Hao Liang, Chengyu Shen, Bin Cui, and Wentao Zhang. 2025. Logicpuzzlerl: Cultivating robust mathematical reasoning in llms via reinforcement learning. *Preprint*, arXiv:2506.04821.

Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, Chenyang Shao, Yuwei Yan, Qinglong Yang, Yiwen Song, Sijian Ren, Xinyuan Hu, Yu Li, Jie Feng, Chen Gao, and Yong Li. 2025. Towards large reasoning models: A survey of reinforced reasoning with large language models. *Preprint*, arXiv:2501.09686.

Nan Xu and Xuezhe Ma. 2024. Llm the genius paradox: A linguistic and math expert's struggle with simple word-based counting problems. *arXiv preprint arXiv:2410.14166*.

Yang Yan, Yu Lu, Renjun Xu, and Zhenzhong Lan. 2025. Do phd-level llms truly grasp elementary addition? probing rule learning vs. memorization in large language models. *Preprint*, arXiv:2504.05262.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025b. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Yao Yao, Zuchao Li, and Hai Zhao. 2023b. Beyond chain-of-thought, effective graph-of-thought reasoning in language models. *arXiv preprint arXiv:2305.16582*.

Yusen Zhang, Sarkar Snigdha Sarathi Das, and Rui Zhang. 2024. Verbosity $\backslash neq$ veracity: Demystify verbosity compensation behavior of large language models. *arXiv preprint arXiv:2411.07858*.

Haoran Zhao, Yuchen Yan, Yongliang Shen, Haolei Xu, Wenqi Zhang, Kaitao Song, Jian Shao, Weiming Lu, Jun Xiao, and Yueting Zhuang. 2025. Let llms break free from overthinking via self-braking tuning. *Preprint*, arXiv:2505.14604.

Contents of the Appendix

A Overthinking Score: Complete Theoretical Framework **12**
- A.1 Fundamental Properties and Proofs 13
- A.2 Sensitivity Analysis and Derivatives 13
- A.3 Hessian Analysis and Concavity . 14
- A.4 Metric Design Rationale and Comparison 14
- A.5 Comparison with Alternative Metrics 14
- A.6 Efficiency Normalization 15
- A.7 Sensitivity to Outliers 15

B Alternative Formulations and Justifications **15**
- B.1 Comparison of Aggregation Functions 15
- B.2 Empirical Comparison 15
- B.3 Weighted Formulations 15

C Mathematical Foundations of Task Suite **16**
- C.1 Basic Arithmetic Operations . . . 16
- C.2 Extremum Detection Tasks 17
- C.3 Statistical Operations 18
- C.4 Counting Operations 18

D Pareto Frontier Analysis **19**
- D.1 Formal Definition 19
- D.2 Computational Complexity 19

E Experimental Configuration and Implementation Details **19**
- E.1 Dataset Generation Parameters . . 19

F Implementation and Algorithmic Details **20**
- F.1 End-to-End Evaluation Pipeline . 20
- F.2 Hierarchical Answer Extraction . 21
- F.3 Complete Generation Algorithm . 21
- F.4 Response Parsing Framework . . . 22
- F.5 Token Counting Strategy 23
- F.6 Empirical Correction Factor . . . 23

G Empirical Analysis and Extended Results **23**
- G.1 Statistical Significance Testing . . 23
- G.2 Cross-Validation Framework . . . 23
- G.3 Performance by Task Category . . 24
- G.4 Error Analysis 24
- G.5 Token Distribution Analysis . . . 24
- G.6 Correlation Analysis 24
- G.7 Computational Complexity Analysis 24
- G.8 Evaluation Scalability 24

H Detailed Case Studies: Reasoning Patterns Analysis **25**
- H.1 Case Study Overview 25
- H.2 Key Insights from Case Studies . . 25
- H.3 Implications for Model Development 25

I Prompt Templates and Task Design **25**
- I.1 Basic Arithmetic Tasks 25
- I.2 Counting Tasks 26
- I.3 Extremum Detection Tasks 27
- I.4 Statistical Tasks 27

J Answer Parsing Framework **27**
- J.1 Design Principles 27
- J.2 Hierarchical Extraction Strategy . 27

K Data Generation Protocol **27**
- K.1 Generation Principles 27
- K.2 Implementation Details 28
- K.3 Iterative Parser Development . . . 28

L Reasoning Budget Analysis: Detailed Results **28**

M Extended Evaluation Results **28**
- M.1 Quantization Effects on Model Performance 28
- M.2 Constrained Token Budget Evaluation 29

N Detailed Case Studies: Reasoning Patterns Analysis **29**
- N.1 Helpful Reasoning: Systematic Decomposition 29
- N.2 Wasteful Overthinking: Redundant Verification 29
- N.3 Extreme Overthinking: Division . 29
- N.4 Additional Case Studies Across Model Families 29

O Pathological Failure Modes **29**
- O.1 Infinite Character Repetition . . . 29
- O.2 Random Token Sequences 29
- O.3 Self-Contradiction Loops 29
- O.4 Implications for Model Development 29

A Overthinking Score: Complete Theoretical Framework

This section provides a comprehensive theoretical analysis of the Overthinking Score, including rigorous proofs of its mathematical properties, sensitivity analysis, and justification for our choice of the harmonic mean formulation.

A.1 Fundamental Properties and Proofs

We provide complete proofs for the properties of the Overthinking Score $\mathcal{O}_i = \frac{2 \cdot A_i \cdot E_{t,i}}{A_i + E_{t,i}}$.

Theorem 1 (Complete Properties of Overthinking Score)**.** *The Overthinking Score $\mathcal{O}_i$ satisfies:*

1. ***Boundedness:*** *$\mathcal{O}_i \in [0, 1]$ for all valid inputs*
2. ***Symmetry:*** *$\mathcal{O}(A, E) = \mathcal{O}(E, A)$*
3. ***Vanishing Limits:*** *$\lim_{E_t \to 0} \mathcal{O} = 0$ and $\lim_{A \to 0} \mathcal{O} = 0$*
4. ***Monotonicity:*** *$\mathcal{O}$ is strictly increasing in both arguments*
5. ***Sublinearity:*** *$\mathcal{O}(A, E) \leq \min(A, E)$*
6. ***Perfection Condition:*** *$\mathcal{O} = 1 \iff A = E = 1$*

Proof. **Property 1 (Boundedness):** Since $A, E \in [0, 1]$, we have:

$$\mathcal{O} = \frac{2AE}{A + E} \leq \frac{2AE}{2\sqrt{AE}} = \sqrt{AE} \leq 1$$

where we used the AM-GM inequality. The lower bound follows from non-negativity of A and E.

Property 2 (Symmetry): Direct from the commutative property:

$$\mathcal{O}(A, E) = \frac{2AE}{A + E} = \frac{2EA}{E + A} = \mathcal{O}(E, A)$$

Property 3 (Vanishing Limits): As $E_t \to 0$:

$$\mathcal{O} = \frac{2AE_t}{A + E_t} = \frac{2A}{A/E_t + 1}$$

Since $A/E_t \to \infty$ as $E_t \to 0$, we have $\mathcal{O} \to 0$. The proof for $A \to 0$ is symmetric.

Property 4 (Monotonicity): Taking the partial derivative with respect to A:

$$\frac{\partial \mathcal{O}}{\partial A} = \frac{2E^2}{(A + E)^2} > 0 \text{ for } E > 0$$

Similarly, $\frac{\partial \mathcal{O}}{\partial E} = \frac{2A^2}{(A+E)^2} > 0$ for $A > 0$.

Property 5 (Sublinearity): By the harmonic-minimum inequality:

$$\begin{aligned} \mathcal{O} &= \frac{2AE}{A + E} && (7) \\ &\leq \frac{2 \min(A, E) \cdot \min(A, E)}{\min(A, E) + \min(A, E)} && (8) \\ &= \min(A, E) && (9) \end{aligned}$$

Property 6 (Perfection Condition): If $\mathcal{O} = 1$, then:

$$\begin{aligned} \frac{2AE}{A + E} = 1 &\implies 2AE = A + E && (10) \\ &\implies (A - E)^2 = 0 && (11) \end{aligned}$$

Therefore $A = E$. Substituting back: $\mathcal{O} = \frac{2A^2}{2A} = A = 1$, so $A = E = 1$.

Conversely, if $A = E = 1$, then $\mathcal{O} = \frac{2 \cdot 1 \cdot 1}{1+1} = 1$. □

A.2 Sensitivity Analysis and Derivatives

Theorem 2 (Sensitivity Properties)**.** *The Overthinking Score exhibits the following sensitivity characteristics:*

$$\frac{\partial \mathcal{O}}{\partial A} = \frac{2E^2}{(A + E)^2} \quad (12)$$

$$\frac{\partial \mathcal{O}}{\partial E} = \frac{2A^2}{(A + E)^2} \quad (13)$$

$$\frac{\partial^2 \mathcal{O}}{\partial A \partial E} = \frac{4AE}{(A + E)^3} \quad (14)$$

Proof. Starting with $\mathcal{O} = \frac{2AE}{A+E}$, we apply the quotient rule:

For $\frac{\partial \mathcal{O}}{\partial A}$:

$$\begin{aligned} \frac{\partial \mathcal{O}}{\partial A} &= \frac{2E(A + E) - 2AE}{(A + E)^2} && (15) \\ &= \frac{2E^2}{(A + E)^2} && (16) \end{aligned}$$

For $\frac{\partial \mathcal{O}}{\partial E}$:

$$\begin{aligned} \frac{\partial \mathcal{O}}{\partial E} &= \frac{2A(A + E) - 2AE}{(A + E)^2} && (17) \\ &= \frac{2A^2}{(A + E)^2} && (18) \end{aligned}$$

For the cross-partial:

$$\begin{aligned} \frac{\partial^2 \mathcal{O}}{\partial A \partial E} &= \frac{\partial}{\partial A} \left(\frac{2A^2}{(A + E)^2} \right) && (19) \\ &= \frac{4A(A + E)^2 - 2A^2 \cdot 2(A + E)}{(A + E)^4} && (20) \\ &= \frac{4AE}{(A + E)^3} && (21) \end{aligned}$$

□

Corollary 1 (Improvement Incentives)**.** *The sensitivity to each component is proportional to the square of the other component, ensuring that improving the weaker dimension yields greater marginal benefit.*

A.3 Hessian Analysis and Concavity

Theorem 3 (Concavity Properties)**.** *The Hessian matrix of the Overthinking Score:*

$$H = \frac{4AE}{(A+E)^3} \begin{bmatrix} -E & A+E \\ A+E & -A \end{bmatrix} \tag{22}$$

has eigenvalues $\lambda_1 = 0$ *and* $\lambda_2 = -\frac{4AE}{(A+E)^2} < 0$*, confirming concavity along the improvement direction.*

Proof. The Hessian matrix elements are:

$$H_{11} = \frac{\partial^2 \mathcal{O}}{\partial A^2} = -\frac{4E^2}{(A+E)^3}$$

$$H_{22} = \frac{\partial^2 \mathcal{O}}{\partial E^2} = -\frac{4A^2}{(A+E)^3}$$

$$H_{12} = H_{21} = \frac{4AE}{(A+E)^3}$$

Factoring out $\frac{4AE}{(A+E)^3}$:

$$H = \frac{4AE}{(A+E)^3} \begin{bmatrix} -E & A+E \\ A+E & -A \end{bmatrix}$$

The characteristic polynomial is:

$$\det(H - \lambda I) = \lambda^2 + \lambda \cdot \frac{4AE}{(A+E)^2} = 0$$

This yields eigenvalues $\lambda_1 = 0$ and $\lambda_2 = -\frac{4AE}{(A+E)^2}$.

The eigenvector for $\lambda_1 = 0$ is $(1,1)^T$, representing the direction of balanced improvement. The negative eigenvalue λ_2 confirms concavity perpendicular to this direction. □

A.4 Metric Design Rationale and Comparison

The Overthinking Score must satisfy several mathematical properties to serve as a meaningful evaluation metric:

Theorem 4 (Metric Properties)**.** *The Overthinking Score* $\mathcal{O} : [0,1]^2 \to [0,1]$ *satisfies:*

1. ***Symmetry:*** $\mathcal{O}(a,e) = \mathcal{O}(e,a)$
2. ***Monotonicity:*** *$\mathcal{O}$ is increasing in both arguments*
3. ***Boundary Conditions:***

$$\mathcal{O}(0,e) = 0 \quad \forall e \in [0,1] \tag{23}$$

$$\mathcal{O}(a,0) = 0 \quad \forall a \in [0,1] \tag{24}$$

$$\mathcal{O}(1,1) = 1 \tag{25}$$

4. ***Continuity:*** *$\mathcal{O}$ is continuous on* $[0,1]^2$
5. ***Concavity:*** *$\mathcal{O}$ is jointly concave*

Proof. For the harmonic mean $\mathcal{O}(a,e) = \frac{2ae}{a+e}$:

1. **Symmetry:** Immediate from commutativity of multiplication and addition.
2. **Monotonicity:**

$$\frac{\partial \mathcal{O}}{\partial a} = \frac{2e^2}{(a+e)^2} > 0 \text{ for } e > 0 \tag{26}$$

   Similarly for $\partial \mathcal{O}/\partial e$.

3. **Boundary Conditions:** Direct substitution verifies all three conditions.
4. **Continuity:** The function is a ratio of continuous functions with non-zero denominator on $(0,1]^2$, extended continuously to boundaries.
5. **Concavity:** The Hessian matrix is:

$$H = \frac{4ae}{(a+e)^3} \begin{bmatrix} -e & a+e \\ a+e & -a \end{bmatrix} \tag{27}$$

   The eigenvalues are $\lambda_1 = 0$ and $\lambda_2 = -\frac{4ae}{(a+e)^2} \leq 0$, confirming negative semi-definiteness.

□

A.5 Comparison with Alternative Metrics

We formally compare different aggregation methods:

Proposition 1 (Penalty for Imbalance)**.** *Define the imbalance penalty as* $P(a,e) = M(a,e) - M(a,a)$ *where* M *is a mean function. For* $a < e$*:*

$$P_{arith}(a,e) = \frac{e-a}{2} \tag{28}$$

$$P_{geo}(a,e) = a(\sqrt{e/a} - 1) \tag{29}$$

$$P_{harm}(a,e) = a\left(1 - \frac{2e}{a+e}\right) = \frac{a(a-e)}{a+e} \tag{30}$$

The harmonic mean penalty grows superlinearly with imbalance, providing stronger incentive for balanced optimization.

A.6 Efficiency Normalization

The token efficiency component requires careful normalization:

Definition 5 (Token Efficiency)**.** *Given token counts $\{t_1, \ldots, t_m\}$ from m models, the efficiency of model i is:*

$$E_i = \begin{cases} 1 & \text{if } t_i = t_{\min} \\ 0 & \text{if } t_i = t_{\max} \text{ and } t_{\max} > t_{\min} \\ 1 - \frac{t_i - t_{\min}}{t_{\max} - t_{\min}} & \text{otherwise} \end{cases} \tag{31}$$

Lemma 1 (Efficiency Properties)**.** *The efficiency function satisfies:*

1. *$E_i \in [0, 1]$ for all models*
2. *E is strictly decreasing in token count*
3. *The transformation is affine, preserving relative distances*

A.7 Sensitivity to Outliers

Theorem 5 (Robustness to Outliers)**.** *Let $\mathcal{O}_{avg}$ be the average Overthinking Score across n tasks. Adding one outlier task with score $\mathcal{O}_{outlier}$ changes the average by at most $1/n$.*

Proof. Let the original average be $\bar{\mathcal{O}} = \frac{1}{n}\sum_{i=1}^{n} \mathcal{O}_i$. Adding an outlier:

$$\bar{\mathcal{O}}_{\text{new}} = \frac{n\bar{\mathcal{O}} + \mathcal{O}_{\text{outlier}}}{n+1} \tag{32}$$

The change is:

$$|\bar{\mathcal{O}}_{\text{new}} - \bar{\mathcal{O}}| = \left| \frac{\mathcal{O}_{\text{outlier}} - \bar{\mathcal{O}}}{n+1} \right| \leq \frac{1}{n+1} \tag{33}$$

since both scores are in $[0, 1]$. □

B Alternative Formulations and Justifications

Having established the fundamental properties of our Overthinking Score, this section provides theoretical justification for our choice of the harmonic mean and compares it against alternative aggregation functions for combining accuracy and efficiency metrics.

B.1 Comparison of Aggregation Functions

We analyze alternative formulations for combining accuracy and efficiency metrics.

Theorem 6 (Optimality of Harmonic Mean)**.** *Among all symmetric, homogeneous means, the harmonic mean maximally penalizes imbalance between components while maintaining smooth differentiability.*

Proof. Consider the generalized mean family:

$$M_p(a, b) = \left(\frac{a^p + b^p}{2} \right)^{1/p}$$

For different values of p:

- $p = 1$: Arithmetic mean $M_1 = \frac{a+b}{2}$
- $p = 0$: Geometric mean $M_0 = \sqrt{ab}$ (by L'Hôpital's rule)
- $p = -1$: Harmonic mean $M_{-1} = \frac{2ab}{a+b}$

The penalty for imbalance can be measured by the second-order cross-derivative:

$$\frac{\partial^2 M_p}{\partial a \partial b} = \frac{p(p-1)(ab)^{p-1}}{2^{1/p}(a^p + b^p)^{2-1/p}}$$

As $p \to -1$, this penalty is maximized among all finite p values. The harmonic mean also maintains smoothness on $(0, \infty)^2$. □

B.2 Empirical Comparison

Scenario	Arithmetic	Geometric	Harmonic
$A = 1.0, E = 0.1$	0.55	0.32	0.18
$A = 0.9, E = 0.3$	0.60	0.52	0.45
$A = 0.6, E = 0.6$	0.60	0.60	0.60
$A = 0.8, E = 0.8$	0.80	0.80	0.80

Table 4: Comparison of different mean formulations for various accuracy-efficiency scenarios.

The harmonic mean provides the strongest penalty for imbalanced performance, aligning with our goal of identifying overthinking behavior.

B.3 Weighted Formulations

We also considered weighted variants analogous to the F_β score:

$$\mathcal{O}_w = \frac{(1 + \beta^2) \cdot A \cdot E}{\beta^2 \cdot A + E}$$

However, this introduces an arbitrary parameter β requiring domain-specific calibration. For $\beta = 1$, we recover the standard harmonic mean.

C Mathematical Foundations of Task Suite

To ensure rigorous evaluation, this section provides comprehensive mathematical foundations for each task in our evaluation framework, including formal definitions, uniqueness proofs, and computational complexity analysis. Each task is designed to have exactly one correct answer, eliminating ambiguity in evaluation.

C.1 Basic Arithmetic Operations

C.1.1 Sorting Task

Definition 6 (Sorting Problem)**.** *Given an input list $L = (x_1, x_2, \ldots, x_n)$ where $x_i \in \mathbb{Z}$ for all $i \in [1, n]$, the sorting function $f_{sort} : \mathbb{Z}^n \to \mathbb{Z}^n$ produces an output list $L' = (x'_1, x'_2, \ldots, x'_n)$ satisfying:*

1. ***Ordering Property:*** $\forall i \in [1, n-1] : x'_i \leq x'_{i+1}$
2. ***Preservation Property:*** *The multiset $\{x'_1, \ldots, x'_n\}$ equals $\{x_1, \ldots, x_n\}$*

Theorem 7 (Uniqueness of Sorting)**.** *For any input list L, there exists exactly one sorted output L' satisfying both properties.*

Proof. Suppose there exist two distinct sorted lists $L'_1 = (a_1, \ldots, a_n)$ and $L'_2 = (b_1, \ldots, b_n)$ both satisfying the sorting properties for input L.

Let k be the first index where $a_k \neq b_k$. Without loss of generality, assume $a_k < b_k$.

Since both lists preserve the multiset, there must exist $j > k$ such that $a_j = b_k$. But this implies $a_k < b_k \leq b_j$ (by ordering property of L'_2).

The element a_k must appear in L'_2 at some position m. Since $a_k < b_k$ and b_k is at position k, we have $m < k$ (by ordering property).

But this contradicts our assumption that k is the first index where the lists differ. Therefore, $L'_1 = L'_2$, proving uniqueness. □

Algorithm 1: Ground Truth Generation for Sorting

Input: List $L = (x_1, \ldots, x_n)$
Output: Sorted list L'
▷ `Use deterministic sorting algorithm`
$L' \leftarrow$ MergeSort(L)
return L'

Computational Complexity: $O(n \log n)$ time, $O(n)$ space

C.1.2 Comparison Task

Definition 7 (Comparison Problem)**.** *Given two integers $a, b \in \mathbb{Z}$, the comparison function $f_{comp} : \mathbb{Z} \times \mathbb{Z} \to \{<, =, >\}$ is defined as:*

$$f_{comp}(a, b) = \begin{cases} < & \text{if } a < b \\ = & \text{if } a = b \\ > & \text{if } a > b \end{cases} \tag{34}$$

Theorem 8 (Uniqueness of Comparison)**.** *For any pair $(a, b) \in \mathbb{Z}^2$, exactly one relation from $\{<, =, >\}$ holds.*

Proof. The integers $\mathbb{Z}$ form a totally ordered set under the standard ordering $\leq$. By the trichotomy property of total orders, for any $a, b \in \mathbb{Z}$, exactly one of the following holds:

1. $a < b$ (equivalently, $a \leq b$ and $a \neq b$)
2. $a = b$
3. $a > b$ (equivalently, $b < a$)

These three cases are mutually exclusive and exhaustive, ensuring unique output. □

Computational Complexity: $O(1)$ time and space

C.1.3 Sum Task

Definition 8 (Sum Problem)**.** *Given a list $L = (x_1, \ldots, x_n)$ where $x_i \in \mathbb{Z}$, the sum function $f_{sum} : \mathbb{Z}^n \to \mathbb{Z}$ is:*

$$f_{sum}(L) = \sum_{i=1}^{n} x_i \tag{35}$$

Theorem 9 (Uniqueness of Sum)**.** *The sum of integers is uniquely determined.*

Proof. Integer addition is a well-defined binary operation $+ : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ that is associative and commutative. By the fundamental theorem of arithmetic and properties of integer rings, the sum $\sum_{i=1}^{n} x_i$ has exactly one value in $\mathbb{Z}$.

Formally, we can prove by induction on n: - Base case ($n = 1$): $f_{\text{sum}}((x_1)) = x_1$, uniquely determined - Inductive step: If $\sum_{i=1}^{k} x_i$ is unique, then $\sum_{i=1}^{k+1} x_i = \left(\sum_{i=1}^{k} x_i\right) + x_{k+1}$ is unique by well-definedness of addition □

Numerical Bounds: For inputs in range $[-R, R]$, the maximum absolute sum is $n \cdot R$, ensuring no overflow for reasonable parameters.

Computational Complexity: $O(n)$ time, $O(1)$ space

C.1.4 Multiplication Task

Definition 9 (Product Problem)**.** *Given a list $L = (x_1, \ldots, x_n)$ where $x_i \in \mathbb{Z}$, the product function $f_{mult} : \mathbb{Z}^n \to \mathbb{Z}$ is:*

$$f_{mult}(L) = \prod_{i=1}^{n} x_i \tag{36}$$

Theorem 10 (Uniqueness of Product)**.** *The product of integers is uniquely determined.*

Proof. Similar to addition, integer multiplication is a well-defined binary operation $\times : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ that is associative and commutative. The product $\prod_{i=1}^{n} x_i$ is uniquely determined by iterative application of this operation. □

Overflow Considerations: For n integers in range $[-R, R]$, the product magnitude can reach R^n. We handle large products using arbitrary precision arithmetic in ground truth generation.

Computational Complexity: $O(n)$ multiplications, but each multiplication may have complexity $O(m^2)$ for m-bit numbers

C.1.5 Division Task

Definition 10 (Division Problem)**.** *Given $a, b \in \mathbb{Z}$ with $b \neq 0$, the division function $f_{div} : \mathbb{Z} \times (\mathbb{Z} \setminus \{0\}) \to \mathbb{Q}$ is:*

$$f_{div}(a, b) = \frac{a}{b} \tag{37}$$

Theorem 11 (Uniqueness of Division)**.** *For any $a \in \mathbb{Z}$ and $b \in \mathbb{Z} \setminus \{0\}$, the quotient a/b is uniquely determined in $\mathbb{Q}$.*

Proof. The rationals $\mathbb{Q}$ form a field, and division by non-zero elements is the multiplicative inverse operation. For any $b \neq 0$, there exists a unique $b^{-1} \in \mathbb{Q}$ such that $b \cdot b^{-1} = 1$. Therefore, $a/b = a \cdot b^{-1}$ is uniquely determined. □

Representation: We represent the result as a decimal with sufficient precision, rounding to 6 decimal places for evaluation.

Computational Complexity: $O(\log |a| + \log |b|)$ for arbitrary precision

C.1.6 Subtraction and Absolute Difference Tasks

Definition 11 (Subtraction Problem)**.** *Given $a, b \in \mathbb{Z}$, the subtraction function $f_{sub} : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ is:*

$$f_{sub}(a, b) = b - a \tag{38}$$

Definition 12 (Absolute Difference Problem)**.** *Given $a, b \in \mathbb{Z}$, the absolute difference function $f_{absdiff} : \mathbb{Z} \times \mathbb{Z} \to \mathbb{N}_0$ is:*

$$f_{absdiff}(a, b) = |a - b| = \begin{cases} a - b & \text{if } a \geq b \\ b - a & \text{if } a < b \end{cases} \tag{39}$$

Theorem 12 (Uniqueness of Subtraction and Absolute Difference)**.** *Both subtraction and absolute difference yield unique results for any integer pair.*

Proof. Subtraction is the inverse operation of addition in the group $(\mathbb{Z}, +)$, hence uniquely defined. The absolute value function $|\cdot| : \mathbb{Z} \to \mathbb{N}_0$ is well-defined, making their composition unique. □

Computational Complexity: $O(1)$ for machine integers

C.2 Extremum Detection Tasks

C.2.1 Maximum and Minimum Finding

Definition 13 (Extremum Problems)**.** *Given a non-empty list $L = (x_1, \ldots, x_n)$ where $x_i \in \mathbb{Z}$ and $n \geq 1$:*

$$f_{max}(L) = \max_{i \in [1,n]} x_i = \max\{x_1, \ldots, x_n\} \tag{40}$$

$$f_{min}(L) = \min_{i \in [1,n]} x_i = \min\{x_1, \ldots, x_n\} \tag{41}$$

Theorem 13 (Existence and Uniqueness of Extrema)**.** *For any finite non-empty set $S \subset \mathbb{Z}$, both $\max S$ and $\min S$ exist and are unique.*

Proof. Since $\mathbb{Z}$ is totally ordered and S is finite and non-empty:

1. **Existence:** Every finite subset of a totally ordered set has a maximum and minimum element (by well-ordering principle for finite sets).

2. **Uniqueness:** Suppose $m_1, m_2 \in S$ are both maxima. Then $m_1 \leq m_2$ (since m_2 is maximum) and $m_2 \leq m_1$ (since m_1 is maximum). By antisymmetry of $\leq$, we have $m_1 = m_2$.

The same argument applies for minima. □

Computational Complexity: $O(n)$ time, $O(1)$ space

Algorithm 2: Linear-Time Extremum Finding

Input: List $L = (x_1, \ldots, x_n)$
Output: Maximum element m

```
m ← x_1
for i = 2 to n do
    if x_i > m then
        m ← x_i
return m
```

C.3 Statistical Operations

C.3.1 Mean (Average) Calculation

Definition 14 (Arithmetic Mean)**.** *For a list $L = (x_1, \ldots, x_n)$ where $x_i \in \mathbb{Z}$, the mean function $f_{mean} : \mathbb{Z}^n \to \mathbb{Q}$ is:*

$$f_{mean}(L) = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{42}$$

Theorem 14 (Uniqueness of Arithmetic Mean)**.** *The arithmetic mean is uniquely determined for any non-empty list.*

Proof. Since the sum $S = \sum_{i=1}^{n} x_i$ is unique (proven earlier) and $n > 0$, the quotient S/n is uniquely determined in $\mathbb{Q}$. □

Numerical Stability: To avoid overflow in intermediate calculations, we can compute:

$$\bar{x} = x_1 + \frac{1}{n}\sum_{i=2}^{n}(x_i - x_1) \tag{43}$$

Computational Complexity: $O(n)$ time, $O(1)$ space

C.3.2 Median Calculation

Definition 15 (Median)**.** *For a list $L = (x_1, \ldots, x_n)$, let $L' = (x'_1, \ldots, x'_n)$ be its sorted version. The median function $f_{median} : \mathbb{Z}^n \to \mathbb{Q}$ is:*

$$f_{median}(L) = \begin{cases} x'_{(n+1)/2} & \text{if } n \text{ is odd} \\ \frac{x'_{n/2} + x'_{n/2+1}}{2} & \text{if } n \text{ is even} \end{cases} \tag{44}$$

Theorem 15 (Uniqueness of Median)**.** *The median is uniquely determined for any list.*

Proof. The sorted list L' is unique (by sorting uniqueness theorem). For odd n, the middle element is uniquely indexed. For even n, the average of the two middle elements is unique since both elements and division are uniquely determined. □

Lemma 2 (Median Minimizes Absolute Deviation)**.** *The median m minimizes $\sum_{i=1}^{n} |x_i - c|$ over all $c \in \mathbb{R}$.*

Proof. Let $g(c) = \sum_{i=1}^{n} |x_i - c|$. Taking the derivative:

$$\frac{dg}{dc} = \sum_{i:x_i > c} 1 - \sum_{i:x_i < c} 1 \tag{45}$$

This equals zero when the number of points above c equals the number below, which occurs at the median. □

Computational Complexity: $O(n \log n)$ time for sorting-based approach, $O(n)$ expected time using quickselect

C.3.3 Mode Calculation

Definition 16 (Mode)**.** *For a list $L = (x_1, \ldots, x_n)$, let $f_L(v) = |\{i : x_i = v\}|$ be the frequency of value v. The mode function $f_{mode} : \mathbb{Z}^n \to \mathcal{P}(\mathbb{Z})$ returns the set:*

$$f_{mode}(L) = \{v \in L : f_L(v) = \max_{u \in L} f_L(u)\} \tag{46}$$

Theorem 16 (Mode Properties)**.**

1. *The mode set is non-empty for any non-empty list*
2. *The mode set may contain multiple elements (multimodal)*
3. *If all elements are distinct, the mode set equals the entire list*

Proof.

1. Since L is finite and non-empty, $\max_{u \in L} f_L(u)$ exists. At least one element achieves this maximum frequency.
2. Multiple distinct values can have the same maximum frequency.
3. If all elements are distinct, then $f_L(v) = 1$ for all $v \in L$, making every element a mode.

□

Computational Complexity: $O(n)$ expected time with hash map, $O(n \log n)$ worst case

C.4 Counting Operations

C.4.1 Even and Odd Counting

Definition 17 (Parity Counting)**.** *For a list $L = (x_1, \ldots, x_n)$ where $x_i \in \mathbb{Z}$:*

$$f_{even}(L) = |\{i : x_i \equiv 0 \pmod 2\}| \tag{47}$$

$$f_{odd}(L) = |\{i : x_i \equiv 1 \pmod 2\}| \tag{48}$$

Algorithm 3: Mode Computation with Frequency Map

Input: List $L = (x_1, \ldots, x_n)$
Output: Set of modes M

```
freq ← empty hash map
for i = 1 to n do
    freq[x_i] ← freq[x_i] + 1
max_freq ← max_v freq[v]
M ← {v : freq[v] = max_freq}
return M
```

Theorem 17 (Parity Count Properties)**.** *For any list L of length n:*

1. $f_{even}(L) + f_{odd}(L) = n$
2. *Both counts are uniquely determined*
3. $0 \leq f_{even}(L), f_{odd}(L) \leq n$

Proof. 1. Every integer has exactly one parity (even or odd), so the counts partition the list.

2. Parity is well-defined: $x \equiv r \pmod 2$ where $r \in \{0, 1\}$ is unique.

3. Bounds follow from the partition property and non-negativity of cardinality.

□

Computational Complexity: $O(n)$ time, $O(1)$ space

D Pareto Frontier Analysis

Complementing our task-specific analysis, this section provides formal analysis of the Pareto frontier for model comparison, establishing theoretical foundations for multi-objective optimization in our evaluation framework.

D.1 Formal Definition

Definition 18 (Pareto Dominance)**.** *Model M_1 Pareto-dominates model M_2 (denoted $M_1 \succ M_2$) if:*

$$A_{M_1} \geq A_{M_2} \tag{49}$$

$$E_{M_1} \geq E_{M_2} \tag{50}$$

with at least one strict inequality.

Definition 19 (Pareto Frontier)**.** *The Pareto frontier $\mathcal{F}$ is the set of all non-dominated models: $\mathcal{F} = \{M_i : \nexists M_j \text{ such that } M_j \succ M_i\}$*

Theorem 18 (Pareto Frontier Existence and Uniqueness)**.** *For any finite set of models $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$, the Pareto frontier $\mathcal{F}$ exists and is unique.*

Proof. **Existence:** Since $\mathcal{M}$ is finite and the dominance relation $\succ$ is a strict partial order, there must exist at least one model not dominated by any other. If not, we would have an infinite chain of dominance, contradicting finiteness.

Uniqueness: The frontier is uniquely determined by the dominance relation. For any model M, either $M \in \mathcal{F}$ (if no model dominates it) or $M \notin \mathcal{F}$ (if some model dominates it). This binary classification is deterministic. □

D.2 Computational Complexity

Theorem 19 (Frontier Computation Complexity)**.** *Computing the Pareto frontier for m models requires $O(m^2)$ comparisons in the worst case and $O(m \log m)$ comparisons in the average case with appropriate data structures.*

Proof. **Naive approach:** For each model, check dominance against all others: $O(m^2)$.

Optimized approach: Sort models by accuracy ($O(m \log m)$), then traverse maintaining efficiency monotonicity. Models with both higher accuracy and efficiency dominate; this can be checked in a single pass ($O(m)$). Total: $O(m \log m)$. □

E Experimental Configuration and Implementation Details

This section provides complete details of our experimental setup, including sampling strategies, hyperparameters, and computational resources used in our large-scale evaluation.

E.1 Dataset Generation Parameters

Our evaluation protocol employs systematic data generation with the following parameters:

Sampling Strategy.

- **Open-source models:** 1,000 datapoints per task ($N = 1000$)
- **Closed-source models:** 100 datapoints per task ($N = 100$) due to financial constraints
- **Value range:** Uniform sampling from $[-1000, 1000]$ for integer values
- **List sizes:** $\mathcal{L} \in \{8, 16, 32, 64\}$ elements for list-based tasks

- **Multiplication tasks:** Product of $\{2, 4, 8\}$ numbers
- **Cross-validation folds:** $F = 3$ folds for statistical robustness

Inference Hyperparameters. All models were evaluated with standardized inference settings to ensure fair comparison:

- **Temperature:** $T = 0.1$ (near-deterministic sampling)
- **Top-p (nucleus sampling):** $p = 0.9$
- **Maximum tokens:** Model-specific maximum allowed context length
  - Standard models: maximum allowed tokens
  - Reasoning models: maximum allowed tokens (unconstrained), 1024 tokens (constrained evaluation)

Computational Resources. Our experiments utilized:

- **GPU Infrastructure:** NVIDIA A100 (80GB) and H100 GPUs for large models
- **Inference Framework:** vLLM (Kwon et al., 2023) for efficient batched inference (3.7× throughput improvement)
- **Fallback Framework:** HuggingFace Transformers (Wolf et al., 2020) for models incompatible with vLLM
- **Total Compute:** Approximately 2,400 GPU hours across 53 models

These parameters were chosen to balance evaluation thoroughness with computational feasibility while maintaining statistical validity through cross-validation.

F Implementation and Algorithmic Details

Building upon our experimental configuration, this section consolidates all implementation-specific details for our evaluation framework, including data generation, response parsing, and token counting strategies.

F.1 End-to-End Evaluation Pipeline

Algorithm 4 describes the complete end-to-end evaluation workflow, from dataset generation through answer extraction and validation.

Algorithm 4: End-to-End Evaluation Pipeline

Input: Model M, Task set $\mathcal{T} = \{T_1, \ldots, T_{14}\}$, Samples per task N
Output: Aggregated results: accuracy, instruction-following, token counts

▷ `Dataset Generation Phase`
foreach *task* $T_k \in \mathcal{T}$ **do**
  $\mathcal{D}_k \leftarrow \emptyset$
  for $i = 1$ **to** N **do**
    $(q_i, a_i^*) \leftarrow$ GenerateInstance(T_k)
    ▷ `Generate problem and ground truth`
    $\mathcal{D}_k \leftarrow \mathcal{D}_k \cup \{(q_i, a_i^*)\}$
▷ `Model Evaluation Phase`
foreach *task* $T_k \in \mathcal{T}$ **do**
  $\text{correct}_k \leftarrow 0$, $\text{total}_k \leftarrow 0$
  $\text{tokens}_k \leftarrow []$, $\text{words}_k \leftarrow []$
  foreach $(q, a^*) \in \mathcal{D}_k$ **do**
    $r \leftarrow M(q)$
    ▷ `Query model with problem`
    ▷ `Extract answer using hierarchical strategy`
    $a \leftarrow$ HierarchicalExtract(r, T_k)
    ▷ `See Algorithm 5`
    ▷ `Validate and score`
    if *Validate*(a, a^*, T_k) **then**
      $\text{correct}_k \leftarrow \text{correct}_k + 1$
    $\text{total}_k \leftarrow \text{total}_k + 1$
    ▷ `Compute verbosity metrics`
    tokens_k.append(CountTokens(r, M))
    words_k.append(CountWords(r))
  ▷ `Aggregate task-level statistics`
  $\text{accuracy}_k \leftarrow \text{correct}_k / \text{total}_k$
  $\text{mean_tokens}_k \leftarrow \text{mean}(\text{tokens}_k)$
  $\text{std_tokens}_k \leftarrow \text{std}(\text{tokens}_k)$
▷ `Cross-task aggregation`
overall_acc $\leftarrow$ mean$(\{\text{accuracy}_k : k \in [14]\})$
overall_tokens $\leftarrow$ mean$(\{\text{mean_tokens}_k : k \in [14]\})$
▷ `Compute Overthinking Score`
$E_t \leftarrow 1 - \frac{\text{overall_tokens} - \text{tokens}_{\min}}{\text{tokens}_{\max} - \text{tokens}_{\min}}$
$O\text{-Score} \leftarrow \frac{2 \cdot \text{overall_acc} \cdot E_t}{\text{overall_acc} + E_t}$
return $\{$overall_acc, overall_tokens, O-Score, per-task metrics$\}$

F.2 Hierarchical Answer Extraction

Algorithm 5 implements our hierarchical answer extraction strategy, which handles diverse output formats across 50+ models.

Algorithm 5: Hierarchical Answer Extraction

Input: Model response r, Task type T_k
Output: Extracted answer a or failure $\perp$

```
if r contains \boxed{...} then
    a ← ExtractBoxed(r)
    ▷ LaTeX boxed notation
else if r contains explicit answer markers then
    a ← ExtractExplicit(r)
    ▷ Patterns like "Answer:", "=", etc.
else if r contains code blocks then
    a ← ExtractCodeOutput(r)
    ▷ Python/code execution results
else
    a ← ExtractFinalLine(r)
    ▷ Last line heuristic
if Validate(a, T_k) then
    return a
    ▷ Type and format validation
return ⊥
▷ Extraction failed
```

The validation step ensures extracted answers match expected types (integer, list, etc.) and filters out input echoes or malformed outputs. For detailed regex patterns and edge case handling, see Section J.

F.3 Complete Generation Algorithm

We describe here the complete procedure for generating test instances. Let τ denote the target task, and let $\Theta = \{N, F, [r_{\min}, r_{\max}], \mathcal{L}, s\}$ denote the generation parameters. The algorithm produces a dataset $\mathcal{D}$ consisting of input–output pairs.

Initialization. We initialize the dataset as $\mathcal{D} \leftarrow \varnothing$. For each fold index $f \in \{1, \ldots, F\}$, we compute a seed

$$s_f = \mathrm{Hash}(s, f, \tau),$$

where the hash function guarantees cryptographic uniqueness. The random number generator is initialized with seed s_f.

Sample generation. For each sample index $i \in \{1, \ldots, N\}$ we construct an input x_i as follows:

- **List-based tasks.** If τ corresponds to an operation over lists (e.g., $\{\text{sort}, \text{sum}, \text{mult}, \text{mean}, \text{median}, \text{mode}, \text{odd}, \text{even}, \text{max}, \text{min}\}$), we first sample a length

$$n \sim \mathrm{Uniform}(\mathcal{L}).$$

  We then generate n independent elements

$$l_j \sim \mathrm{Uniform}(r_{\min}, r_{\max}), \quad j = 1, \ldots, n,$$

  and form the list $\mathcal{L}_i = [l_1, \ldots, l_n]$. The instance input is $x_i = \mathcal{L}_i$.

- **Binary tasks.** If τ corresponds to a binary operation (e.g., comparison, division), we instead draw

$$a \sim \mathrm{Uniform}(r_{\min}, r_{\max}),$$
$$b \sim \mathrm{Uniform}(r_{\min}, r_{\max})$$

  and set $x_i = (a, b)$. Task-specific adjustments are then applied:

  1. If τ = division, we enforce $b \neq 0$ by resampling b until this condition is satisfied.
  2. If τ = comparison, we explicitly enforce diversity by overwriting (a, b) with one of three configurations with fixed probabilities: $a = b$ (probability 0.33), $a < b$ (probability 0.33), or $a > b$ (probability 0.34).

Label computation and validation. For each constructed input x_i, we compute the ground truth label

$$y_i = f_\tau(x_i).$$

If the validation function $\mathrm{ValidateInput}(x_i, y_i, \tau)$ returns true, we augment the dataset:

$$\mathcal{D} \leftarrow \mathcal{D} \cup \{(x_i, y_i)\}.$$

Otherwise, the sample is discarded and regenerated.

Output. After all folds and samples are processed, the procedure returns the dataset $\mathcal{D}$.

F.4 Response Parsing Framework

F.4.1 Complete Parsing Procedure

Given a raw model response r, the task τ, and input x, the parsing system extracts an answer $\hat{y}$ or returns $\perp$ if no valid answer is found. The method applies a hierarchy of increasingly permissive strategies:

1. **Boxed expressions.** If r contains LaTeX-style boxed patterns of the form $\backslash$boxed$\{\cdot\}$, the last such match is extracted and cleaned, and returned if it validates.

2. **Explicit answer markers.** If r contains markers such as "answer is", "final answer", or "solution:", the text following the marker is extracted, cleaned, and validated.

3. **Code blocks.** If r contains markdown code blocks, these are scanned in reverse order to locate output markers. The last returned or printed value is extracted.

4. **Task-specific validation.** Depending on τ, additional constraints are enforced:

   - Sorting: the parsed $\hat{y}$ must be a permutation of x and sorted in non-decreasing order.
   - Arithmetic tasks (sum, multiplication): $\hat{y}$ must be an integer not equal to any element of x.
   - Division: $\hat{y}$ must be rational and within tolerance ϵ of the true result.
   - Comparison: $\hat{y} \in \{>, <, =\}$.
   - Mean/median: $\hat{y} \in \mathbb{Q}$ and not equal to any element of x.
   - Mode: $\hat{y}$ must be an element of x achieving maximum frequency.
   - Odd/even counts: $\hat{y} \in \mathbb{N} \cup \{0\}$ and bounded by $|x|$.
   - Max/min: $\hat{y}$ must be one of the elements of x.

5. **Final line heuristic.** If all else fails, the system extracts numerical values from the last lines of r and accepts the first one passing validation.

If no stage succeeds, the parser outputs $\perp$.

F.4.2 Validation Functions

Validation routines are task-specific:

- *Sorting:* check that $\hat{y}$ is a sequence with the same multiset of elements as x and non-decreasing order.
- *Comparison:* map textual variants ("greater than", "less than", "equal to") to canonical symbols.
- *Arithmetic:* extract a numerical value and discard if it simply echoes an element of x.
- *Default:* accept the candidate.

F.4.3 Parser Design Principles

Our parsing system must handle diverse output formats while maintaining correctness. We establish formal criteria for valid parsing:

Definition 20 (Valid Parser)**.** *A parser $\mathcal{P} : \mathcal{R} \to \mathcal{C} \cup \{\perp\}$ is valid for task T if:*

1. ***Soundness:*** *If $\mathcal{P}(r) = c$ and $c \neq \perp$, then c represents the model's intended answer*
2. ***Completeness:*** *If response r contains a clear answer c, then $\mathcal{P}(r) \neq \perp$*
3. ***Determinism:*** *$\mathcal{P}$ is a function (same input yields same output)*

F.4.4 Parser Evaluation and Validation

To validate our parsing framework, we manually evaluated a stratified random sample of 500 model responses across different tasks and model families. Table 5 presents extraction success rates by parsing strategy.

The high success rate (98.7%) validates our hierarchical approach, where primary strategies (boxed extraction) handle the majority of cases with high precision, while fallback strategies cover edge cases with acceptable accuracy. The low false positive rate (0.9%) ensures extracted answers reliably represent model outputs rather than parsing artifacts.

Error Analysis. Manual inspection of the 1.3% parsing failures reveals three main categories: **(1)** Ambiguous formats where models provide multiple contradictory answers (47% of failures); **(2)** Extreme verbosity where answers are buried in thousands of tokens without clear markers (31%); and **(3)** Pathological outputs like infinite character repetition (22%). These failure modes represent

Parsing Strategy	Success Rate	False Pos.	Coverage	Latency (ms)
Primary: Boxed extraction	94.2%	0.3%	68.4%	1.2
Secondary: Answer markers	88.7%	1.8%	22.3%	2.1
Tertiary: Code blocks	85.3%	2.4%	5.1%	3.8
Fallback: Last-line heuristic	72.1%	8.7%	4.2%	0.8
Overall (hierarchical)	**98.7%**	**0.9%**	100%	1.8

Table 5: Parsing framework evaluation across 500 manually-validated responses. **Success Rate:** Percentage of responses where extracted answer matches ground truth. **False Pos.:** Rate of extracting incorrect answers. **Coverage:** Percentage of responses handled by each strategy. **Latency:** Average parsing time per response. The hierarchical approach achieves 98.7% overall success with only 0.9% false positives.

genuine model issues rather than parser limitations, supporting our decision to exclude unparseable responses from accuracy calculations.

F.5 Token Counting Strategy

Our hierarchical token counting approach ensures accurate measurement across diverse model families:

1. **Model-specific tokenizers:** For open models, we use the exact tokenizer:
   - Llama models: LlamaTokenizer
   - GPT models: tiktoken with appropriate encoding
   - Mistral/Mixtral: MistralTokenizer
2. **Fallback strategies:**
   - Closed models: tiktoken cl100k_base encoding
   - Unknown models: Whitespace splitting with correction factor $\alpha = 1.33$

F.6 Empirical Correction Factor

We derived the correction factor $\alpha = 1.33$ empirically:

$$\alpha = \frac{1}{N} \sum_{i=1}^{N} \frac{\text{TokenCount}_{\text{actual}}(s_i)}{\text{WordCount}(s_i)} \tag{51}$$

where we sampled $N = 10,000$ responses across different models and computed the average ratio of actual tokens to whitespace-separated words.

G Empirical Analysis and Extended Results

Building upon our theoretical framework and implementation details, this section presents comprehensive empirical validation of our approach, including statistical significance testing, detailed performance breakdowns, and correlation analysis across different model characteristics.

G.1 Statistical Significance Testing

To validate that observed differences in Overthinking Scores are statistically significant, we employ paired t-tests with Bonferroni correction:

Definition 21 (Significance Test)**.** *For models M_1, M_2 evaluated on n tasks, let $d_i = \mathcal{O}_{M_1,i} - \mathcal{O}_{M_2,i}$ be the score difference on task i. The test statistic is:*

$$t = \frac{\bar{d}}{SE(\bar{d})} = \frac{\bar{d}}{s_d/\sqrt{n}} \tag{52}$$

where s_d is the sample standard deviation of differences.

With m models, we perform $\binom{m}{2}$ pairwise comparisons, requiring Bonferroni-adjusted significance level $\alpha' = \alpha/\binom{m}{2}$.

G.2 Cross-Validation Framework

We employ k-fold cross-validation to ensure robustness:

Algorithm 6: K-Fold Evaluation Protocol

Input: Model M, Task set $\mathcal{T}$, Folds k
Output: Mean and variance of Overthinking Score

Partition data into k equal folds $F_1, \ldots, F_k$
for $i = 1$ *to* k **do**
  Evaluate M on fold F_i
  Compute $\mathcal{O}_i$ for fold i
$\bar{\mathcal{O}} \leftarrow \frac{1}{k}\sum_{i=1}^{k} \mathcal{O}_i$
$\sigma^2 \leftarrow \frac{1}{k-1}\sum_{i=1}^{k}(\mathcal{O}_i - \bar{\mathcal{O}})^2$
return $(\bar{\mathcal{O}}, \sigma^2)$

Category	Avg Accuracy	Avg Tokens	Avg $\mathcal{O}$ Score
Basic Arithmetic	72.3%	487	0.621
Extremum Detection	81.2%	342	0.698
Statistical	64.7%	623	0.542
Counting	77.9%	398	0.654

Table 6: Performance breakdown by task category across all models

G.3 Performance by Task Category

We analyze performance patterns across task categories:

Statistical tasks show lowest performance despite being computationally simple, suggesting models struggle with multi-step procedures even when each step is elementary.

G.4 Error Analysis

We categorize errors into systematic patterns:

Definition 22 (Error Taxonomy)**.** *Model errors fall into five categories:*

1. ***Computational Errors:*** *Incorrect arithmetic operations*
2. ***Procedural Errors:*** *Wrong algorithm or step sequence*
3. ***Format Errors:*** *Correct answer in wrong format*
4. ***Partial Errors:*** *Incomplete computation*
5. ***Hallucination:*** *Completely unrelated output*

Distribution of errors reveals that reasoning models predominantly suffer from computational errors (47%) despite lengthy explanations, while standard models show more format errors (31%) but fewer computational mistakes.

G.5 Token Distribution Analysis

The distribution of token lengths follows distinct patterns:

Proposition 2 (Token Length Distribution)**.** *For standard models, token lengths follow approximately log-normal distribution:*

$$f(x; \mu, \sigma) = \frac{1}{x\sigma\sqrt{2\pi}} \exp\left(-\frac{(\ln x - \mu)^2}{2\sigma^2}\right) \tag{53}$$

while reasoning models show bimodal distribution with peaks at constraint boundaries.

This suggests reasoning models have learned specific response templates rather than adaptive generation strategies.

G.6 Correlation Analysis

We examine relationships between model characteristics and overthinking:

Definition 23 (Correlation Metrics)**.** *For model characteristic X (e.g., parameter count) and Overthinking Score $\mathcal{O}$:*

$$\rho_{X,\mathcal{O}} = \frac{Cov(X, \mathcal{O})}{\sigma_X \sigma_{\mathcal{O}}} \tag{54}$$

Our analysis reveals:

- **Size-Score Correlation:** $\rho_{\text{size},\mathcal{O}} = -0.23$ (weak negative)
- **Instruction-Score Correlation:** $\rho_{\text{instruct},\mathcal{O}} = 0.42$ (moderate positive)
- **Reasoning-Score Correlation:** $\rho_{\text{reasoning},\mathcal{O}} = -0.67$ (strong negative)

G.7 Computational Complexity Analysis

G.7.1 Task Complexity

Each task has well-defined computational complexity:

Task	Time Complexity	Space Complexity
Sorting	$O(n \log n)$	$O(n)$
Sum/Product	$O(n)$	$O(1)$
Mean/Mode	$O(n)$	$O(n)$
Median	$O(n \log n)$	$O(1)$
Min/Max	$O(n)$	$O(1)$
Comparison	$O(1)$	$O(1)$
Division	$O(\log a + \log b)$	$O(1)$
Count Even/Odd	$O(n)$	$O(1)$

Table 7: Computational complexity of ground truth generation

G.8 Evaluation Scalability

The total evaluation complexity for m models on n problems with average response length ℓ is:

$$\mathcal{C}_{\text{total}} = O(m \cdot n \cdot (\mathcal{C}_{\text{inference}} + \mathcal{C}_{\text{parse}} + \mathcal{C}_{\text{validate}})) \tag{55}$$

where:

- $\mathcal{C}_{\text{inference}} = O(\ell)$ for token generation
- $\mathcal{C}_{\text{parse}} = O(\ell)$ for response parsing
- $\mathcal{C}_{\text{validate}} = O(k)$ for answer validation (task-dependent)

H Detailed Case Studies: Reasoning Patterns Analysis

This section provides comprehensive case studies examining when reasoning chains help versus hurt model performance. We present detailed examples from Phi-4 (concise) and Phi-4-reasoning (verbose) models, analyzing their reasoning strategies, token efficiency, and accuracy patterns across various task types.

H.1 Case Study Overview

Table 8 presents six representative examples spanning simple arithmetic, complex multi-step operations, and edge cases. For each example, we show the complete model responses, token counts, and reasoning pattern analysis.

H.2 Key Insights from Case Studies

Pattern 1: Redundant Verification Loops On simple arithmetic (sum, division, mean), Phi-4-reasoning systematically over-verifies. The model computes correctly on first attempt but generates 2–4 additional verification passes using the same or slightly varied methods. This adds 17–31× tokens with **zero accuracy improvement**.

Pattern 2: Helpful Structured Decomposition For complex tasks (sorting 9 elements, multi-digit multiplication), explicit decomposition sometimes aids correctness. Phi-4-reasoning's systematic approach (separate negative/positive subgroups, sort each, merge) provides verifiable intermediate steps. However, even here, the token cost is 3–12×, raising questions about efficiency.

Pattern 3: Inability to Adapt The most striking finding: Phi-4-reasoning applies the *same verbose pattern* regardless of task complexity. It cannot distinguish a trivial division (984÷24) from a complex sorting task, wasting tokens uniformly. This suggests lack of **metacognitive control**—models cannot assess problem difficulty or modulate reasoning depth accordingly.

Pattern 4: Error Introduction in Long Chains Counter-intuitively, longer reasoning sometimes *increases* error risk. In the median example, Phi-4-reasoning's verbose exploration introduces confusion (miscounting elements, second-guessing odd-/even), requiring additional tokens to self-correct. While it eventually arrives at the correct answer, the path is error-prone and inefficient.

Pattern 5: Token Budget Brittleness When constrained to 1,024 tokens, Phi-4-reasoning fails to prioritize essential computation. It begins with its standard verbose template (problem restatement, method enumeration) and runs out of tokens before completing verification. This causes the catastrophic accuracy collapse documented in §4.3 of the main paper.

H.3 Implications for Model Development

These case studies reveal that current reasoning models lack two critical capabilities:

1. **Adaptive Computation:** Models cannot modulate reasoning depth based on task complexity. A well-designed system would use minimal tokens for simple operations and allocate budget to genuinely complex tasks.

2. **Termination Criteria:** Models lack confidence-based stopping. They cannot assess when sufficient verification has been achieved, leading to infinite-loop-like verification patterns.

Future work should focus on training models with *step-level value functions* that assess marginal utility of additional reasoning tokens, enabling adaptive and efficient computation allocation.

I Prompt Templates and Task Design

To ensure clarity and reproducibility, we present the complete prompt templates used for each task. Each prompt is designed to be concise yet explicit, clearly stating the task objective, providing the input, and specifying the required output format. For tasks where computation depends on list size (e.g., median calculation with even versus odd lengths), we include guidance on handling different scenarios. All prompts instruct models to present their final answer in the `\boxed{}` format to facilitate consistent and reliable parsing.

I.1 Basic Arithmetic Tasks

Prompt 1: Sum Calculation

Prompt: Add the following list of numbers: *{data_point}*
Provide the sum. Your final answer must be in the format \boxed{answer} at the end.

	Phi-4				Phi-4-reasoning			
Task Type	**Model**	**Response Excerpt**	**Analysis**	**Tokens**	**Model**	**Response Excerpt**	**Analysis**	**Tokens**
Sum (Simple)	Phi-4	"We add step by step: 309 + (-772) = -463, -463 + (-949) = -1412, -1412 + 518 = -894, -894 + (-437) = -1331, -1331 + (-499) = -1830, -1830 + (-543) = -2373, -2373 + (-715) = -3088. $\boxed{-3088}$".	**Efficient:** Direct sequential addition with clear intermediate steps. Each step verifiable.	272	Phi-4-reasoning	"<think>We need to sum [309, -772, -949, 518, -437, -499, -543, -715]. Let's sum positive: 309 + 518 = 827. Negative: -772 - 949 = -1721, -1721 - 437 = -2158, -2158 - 499 = -2657, -2657 - 543 = -3200, -3200 - 715 = -3915. Total = 827 - 3915 = -3088. *Double-check*: 309 - 772 = -463, -463 - 949 = -1412, -1412 + 518 = -894...*[repeats entire calculation 3 more times with slight variations]*. Final: $\boxed{-3088}$".	**Wasteful:** Redundant verification. Computes correctly first time but continues with 3 additional verification passes. No accuracy benefit.	4,837
Sorting (Complex)	Phi-4	"Sorted: [-912, -789, -234, -156, 423, 445, 601, 778, 892]. $\boxed{[-912, -789, -234, -156, 423, 445, 601, 778, 8}$	**Direct:** Immediate answer with correct ordering. Minimal tokens.	89	Phi-4-reasoning	"<think>Identify negatives: -156, -789, -234, -912. Sort: -912 < -789 < -234 < -156. Positives: 892, 423, 601, 778, 445. Sort: 423 < 445 < 601 < 778 < 892. Merge: [-912, -789, -234, -156, 423, 445, 601, 778, 892]. $\boxed{[-912, \ldots, 892]}$".	**Helpful:** Systematic decomposition (separate negative/-positive, sort each, merge) aids correctness on complex task. Structured approach justifies token cost.	312
Division	Phi-4	"984 ÷ 24: 24 × 40 = 960, remainder = 24. 24 ÷ 24 = 1. Answer: 40 + 1 = 41. $\boxed{41}$".	**Concise:** Minimal verification, direct computation.	47	Phi-4-reasoning	"<think>984 ÷ 24. Try: 24 × 40 = 960, remainder 24. So 40 + 1 = 41. *Wait, check*: 24 × 41 = 984? Let me verify: 24 × 40 = 960, 24 × 1 = 24, sum = 984. *Correct*. But let me recalculate using long division...*[performs long division step-by-step]*. Confirms 41. *Triple-check using estimation*... $\boxed{41}$".	**Harmful:** Excessive verification (algebraic, long division, estimation) for simple task. Introduces intermediate doubts but arrives at same answer. Pure overhead.	1,456
Multiplication	Phi-4	"347 × 28: Break down: 347 × 20 = 6,940, 347 × 8 = 2,776. Sum: 6,940 + 2,776 = 9,716. $\boxed{9716}$".	**Structured:** Uses decomposition appropriately for two-digit multiplication.	98	Phi-4-reasoning	"<think>347 × 28. Method 1: 347 × 20 = 6,940, 347 × 8 = 2,776, sum = 9,716. *Verify with Method 2*: 300 × 28 = 8,400, 47 × 28 = 1,316, sum = 9,716. *Method 3 (standard algorithm): [shows vertical multiplication]*. All methods agree: $\boxed{9716}$".	**Mixed:** Multiple verification methods confirm answer but consume 12× tokens. Some redundancy, but cross-validation could catch errors in principle.	1,189
Mean (Edge)	Phi-4	"Sum = -3,630, count = 8, mean = -3630/8 = -453.75. $\boxed{-453.75}$".	**Direct:** Correct handling of negative mean.	41	Phi-4-reasoning	"<think>Sum [-136, -935, -939, -809, -553, -524, 34, 232]. Sum = -3,630. Count = 8. Mean = -3,630 ÷ 8. Let me compute: -3,630 ÷ 8 = -453.75. *Verify*: -453.75 × 8 = -3,630? Check: -453.75 × 8 = -3,630. *Correct*. $\boxed{-453.75}$".	**Moderate:** Some verification justified for negative result sanity check, but still verbose (8× tokens).	327
Median (Odd)	Phi-4	"Sort: [34, 116, 209, 385, 508, 516, 827]. Middle = 385. $\boxed{385}$".	**Minimal:** Correct median extraction.	38	Phi-4-reasoning	"<think>List: [508, -791, 385, 516, 827, 116, -822, 209]. Sort: [-822, -791, 116, 209, 385, 508, 516, 827]. *Wait, recount*: 8 elements (even), not odd. Middle two: 209, 385. Median = (209 + 385)/2 = 297. *But checking again...[recounts, discovers error, corrects to 7 elements]*: Actually [508, 385, 516, 827, 116, 209] after removing negatives? *No, keep all*. Median of 8 = 297. $\boxed{297}$".	**Error-prone:** Verbose reasoning introduces confusion and errors. Eventually self-corrects but at high token cost. Shows brittleness of long chains.	892

Table 8: Detailed case studies across task types. Colors indicate: Yellow = Wasteful overthinking, Red = Helpful decomposition, Green = Harmful verbosity, Blue = Mixed benefit, Purple = Moderate overhead, Pale yellow = Error-prone verbosity.

Prompt 2: Sorting

Prompt: Sort the following list of numbers in ascending order:
{data_point}
Provide the sorted list. Your final answer must be in the format \boxed{answer} at the end.

Prompt 3: Number Comparison

Prompt: Compare the following two numbers and determine their relationship:
Number 1: *{num1}*
Number 2: *{num2}*
Is Number 1 greater than, less than, or equal to Number 2? Your final answer must be in the format \boxed{relation} at the end, where 'relation' is one of: 'greater than', 'less than', or 'equal to'.

Prompt 4: Subtraction Calculation

Prompt: Subtract *{num1}* from *{num2}* and provide your final answer in \boxed{answer} format at the end of your response.

Prompt 5: Absolute Difference Calculation

Prompt: Find the absolute difference between the following two numbers:
Number 1: *{num1}*, Number 2: *{num2}*
Provide the result. Your final answer must be in the format \boxed{answer} at the end.

Prompt 6: Multiplication Calculation

Prompt: Multiply the following list of numbers:
{data_point}
Provide the product. Your final answer must be in the format \boxed{answer} at the end.

Prompt 7: Division Calculation

Prompt: Divide *{num1}* by *{num2}*
Provide the answer as a floating point number. Your final answer must be in the format \boxed{answer} at the end.

I.2 Counting Tasks

Prompt 8: Even Count

Prompt: Count the even numbers from the following list of numbers:
{data_point}
Provide the final count of even numbers. Your final answer must be in the format \boxed{answer} at the end.

Prompt 9: Odd Count

Prompt: Count the odd numbers from the following list of numbers:
{data_point}
Provide the final count of odd numbers. Your final answer must be in the format \boxed{answer} at the end.

I.3 Extremum Detection Tasks

Prompt 10: Find Minimum

Prompt: Find the minimum number from the given list of numbers. List = *{data_point}*. Your final answer must be in the format \boxed{minimum} at the end of your response.

Prompt 11: Find Maximum

Prompt: Find the maximum number from the given list of numbers. List = *{data_point}*. Your final answer must be in the format \boxed{maximum} at the end of your response.

I.4 Statistical Tasks

Prompt 12: Mean Calculation

Prompt: Calculate the mean (average) of the following list of numbers:
{input_list}
The mean is the sum of all numbers divided by the count of numbers. Calculate the exact mean value. Your final answer must be in the format \boxed{mean value} at the end.

Prompt 13: Median Calculation

Prompt: Find the median value of the following list of numbers:
{input_list}
The median is the middle value when the list is sorted. If there is an even number of elements, the median is the average of the two middle values. Your final answer must be in the format \boxed{median value} at the end.

Prompt 14: Mode Calculation

Prompt: Find the mode(s) of the following list of numbers:
{input_list}
The mode is the value that appears most frequently. If multiple values appear with the same highest frequency, return all of them. Your final answer must be in the format \boxed{mode(s)} at the end. If there are multiple modes, list them separated by commas.

J Answer Parsing Framework

Given the variability in model response formats, ranging from concise answers to verbose multi-step reasoning with inconsistent formatting, a robust parsing architecture is essential.

J.1 Design Principles

Our parsing framework addresses several key challenges:

1. **Format Diversity:** Models produce answers in various formats (boxed notation, explicit markers, code blocks, plain text)
2. **Embedded Context:** Responses often include input restatement, intermediate steps, and explanatory text
3. **Numerical Variations:** Numbers appear in different notations (scientific, fractional, decimal)
4. **False Positives:** Input values must be distinguished from computed results

J.2 Hierarchical Extraction Strategy

The parser implements a multi-layered approach with task-specific validation:

1. **Primary Layer:** Extract `\boxed{}` patterns using regex matching
2. **Secondary Layer:** Parse explicit answer markers ("The answer is...", "Final answer:")
3. **Tertiary Layer:** Extract from code blocks or markdown formatting
4. **Fallback Layer:** Apply task-specific heuristics to final lines
5. **Validation:** Ensure extracted value represents solution, not input echo

This hierarchical design achieved 98.7% successful extraction across 2.1 million model inferences, minimizing false negatives while maintaining precision.

K Data Generation Protocol

Dynamic test generation is central to our evaluation methodology. Unlike static benchmarks, on-the-fly generation ensures that models cannot rely on memorized training examples, providing more accurate assessments of genuine computational ability.

K.1 Generation Principles

Our generation protocol satisfies several key properties:

1. **Contamination Resistance:** New instances generated per evaluation prevent memorization
2. **Reproducibility:** Cryptographic seeding enables exact replication across runs
3. **Controlled Difficulty:** Parameterized ranges and sizes allow systematic testing
4. **Task-Specific Constraints:** Domain requirements enforced (e.g., non-zero denominators for division)

K.2 Implementation Details

For each task and fold combination, we:

1. Initialize random number generator with deterministic seed $s_f = \text{Hash}(s, f, \tau)$
2. Sample list lengths from $\mathcal{L} = \{8, 16, 32, 64\}$ for list-based tasks
3. Draw values from Uniform$[-1000, 1000]$ ensuring numerical diversity
4. Apply task-specific validation and constraints
5. Compute and store ground truth for verification

This approach generated 42,000 unique test instances per model (1,000 samples × 3 folds × 14 tasks), totaling over 2.1 million inferences across our 50+ model evaluation.

K.3 Iterative Parser Development

The parsing system underwent iterative refinement based on empirical analysis. Initial rules were designed from common patterns, but as evaluation expanded to 40+ models, we encountered diverse edge cases. We systematically reviewed misparsed outputs, categorizing failure modes (input echo, non-standard syntax, ambiguous formatting). Each iteration added targeted regular expressions, validation checks, and fallback strategies, evaluated against curated challenging cases. This continuous refinement ensured robustness across model families without overfitting to specific output styles.

L Reasoning Budget Analysis: Detailed Results

This section presents comprehensive results from our reasoning budget experiments across Gemini, GPT-5, and O-series models. Table 9 shows performance across different budget configurations, revealing how models respond to varying computational allocations.

Key observations from the detailed budget analysis:

- **Gemini models** show modest gains with increased budgets, with Gemini-2.5-Pro achieving only 2% absolute improvement from disabled to maximum budget despite 80× token increase.
- **GPT-5 family** demonstrates plateau behavior, where GPT-5 reaches 97% accuracy at medium effort and shows zero improvement at high effort, indicating optimal performance at moderate budgets.
- **O-series models** maintain remarkably stable accuracy across all budget levels (O3 at 97%, O3-Mini at 93%, O4-Mini at 95%), suggesting these models have converged to their capability ceiling on basic arithmetic regardless of additional computation.
- **Overthinking Score** reveals that higher budgets do not necessarily improve the efficiency-accuracy tradeoff, with many models achieving best O-Scores at lower or medium budgets.

M Extended Evaluation Results

This section presents comprehensive results from our quantization and constrained generation experiments, providing detailed performance breakdowns that complement the main paper findings.

M.1 Quantization Effects on Model Performance

Table 10 shows the impact of GPTQ quantization (8-bit and 4-bit) on the Qwen2.5 model family across different scales. These results demonstrate that larger models exhibit greater robustness to aggressive quantization, while smaller models suffer more significant performance degradation.

Key findings from quantization experiments:

- **Size-dependent robustness:** Models ≥32B parameters retain >99% of full-precision accuracy even at 4-bit quantization

- **Small model vulnerability:** The 0.5B model suffers 40% relative accuracy loss at 4-bit (21.31% → 12.77%)
- **Mid-size trade-offs:** 7B-14B models show modest 3-5% relative degradation at 4-bit
- **Overthinking Score stability:** Quantization minimally impacts the Overthinking Score, suggesting behavioral patterns persist across precision levels

M.2 Constrained Token Budget Evaluation

Table 11 presents results from our constrained generation experiments, where models were limited to 1024 output tokens. This setting reveals how reasoning models cope with restricted thinking budgets.

Critical observations from constrained experiments:

- **Catastrophic degradation in reasoning models:** Phi-4-reasoning accuracy drops from 72.23% to 53.48% (-26% relative) under 1024-token constraint
- **Non-adaptive generation:** Even when constrained, reasoning models use $\sim$1000 tokens on average, suggesting fixed exploration patterns rather than problem-adaptive computation
- **Qwen3 family consistency:** All Qwen3 models show similar degradation patterns, losing $\sim$60% accuracy under constraint regardless of size
- **Overthinking Score collapse:** Constrained reasoning models achieve Overthinking Scores below 0.20, indicating severe imbalance between accuracy and efficiency

N Detailed Case Studies: Reasoning Patterns Analysis

This section provides in-depth analysis of reasoning patterns across different models, examining both successful and failed attempts to understand what distinguishes helpful reasoning from wasteful verbosity.

N.1 Helpful Reasoning: Systematic Decomposition

For complex tasks like sorting, structured reasoning can genuinely aid correctness:

N.2 Wasteful Overthinking: Redundant Verification

Simple arithmetic reveals harmful verbosity where models verify correct answers repeatedly:

N.3 Extreme Overthinking: Division

The most extreme overthinking occurs in simple division where models use 31× more tokens:

N.4 Additional Case Studies Across Model Families

Beyond the Phi-4 family, other model families exhibit similar patterns of overthinking, revealing that the phenomenon is widespread across different architectures and training paradigms.

N.4.1 Llama Models: Size vs Efficiency Trade-off

N.4.2 Qwen Models: Constrained Generation Reveals Dependence

N.4.3 GPT-4o Models: Minimal vs Extended Reasoning

N.4.4 O-series Models: Extended Budget Without Accuracy Gain

O Pathological Failure Modes

Beyond inefficient reasoning, some models exhibit catastrophic failure modes that reveal fundamental issues in generation control and stopping criteria.

O.1 Infinite Character Repetition

Some reasoning models, when uncertain, fall into infinite loops generating the same character:

O.2 Random Token Sequences

Another pathological pattern involves generation of semantically meaningless random characters:

O.3 Self-Contradiction Loops

Extended reasoning can lead to models contradicting their own correct intermediate results:

O.4 Implications for Model Development

These pathological cases reveal several architectural deficiencies:

- **Stopping criteria failures:** Models lack robust mechanisms to terminate generation after producing correct, complete answers
- **Token budget mismatch:** Training distributions don't match deployment constraints,

Case Study 1: Successful Sorting with Phi-4-reasoning

Task: Sort the list [847, -123, 456, -789, 234, 901, -456, 123]

Phi-4 Response (89 tokens):

```
The sorted list is:
[-789, -456, -123, 123,
 234, 456, 847, 901]
\boxed{[-789, -456, -123,
123, 234, 456, 847, 901]}
```

Phi-4-reasoning Response (312 tokens):

```
Let me sort step by step:
1. Separate negatives/positives
   Neg: -123, -789, -456
   Pos: 847, 456, 234, 901, 123
2. Sort negatives
   -789 < -456 < -123
3. Sort positives
   123 < 234 < 456 < 847 < 901
4. Combine
\boxed{[-789,-456,-123,123,
234,456,847,901]}
```

Analysis: Both models produce correct answers. The reasoning model uses 3.5× more tokens, but the systematic decomposition (separating negatives/positives, sorting each group) provides verifiable intermediate steps that reduce error risk for complex inputs.

causing aberrant behavior when output length expectations are violated

- **Confidence calibration issues:** Models can't distinguish between "answer found" and "continue exploring" states, leading to unnecessary extended generation
- **Verification without termination:** Models learn to verify answers but not to stop after successful verification, continuing to generate indefinitely

Future work should focus on reward shaping that explicitly penalizes generation beyond answer completeness, adaptive stopping mechanisms that detect when continuation provides diminishing returns, and better calibration of end-of-sequence token probabilities during training.

Case Study 2: Wasteful Overthinking in Sum

Task: Calculate the sum of [45, 78, 23, 91, 67, 34, 89, 12]

Phi-4 Response (272 tokens):

```
Let me add these numbers:
45 + 78 = 123
123 + 23 = 146
146 + 91 = 237
237 + 67 = 304
304 + 34 = 338
338 + 89 = 427
427 + 12 = 439

\boxed{439}
```

Phi-4-reasoning Response (4,837 tokens):

```
Step by step:
First: 45+78=123
Verify: 40+70=110, 5+8=13
Total=123 [OK]

Second: 123+23
Break: 123+20=143, 143+3=146
Verify: 100+23=146 [OK]

... [each step verified]

Re-verify entire chain:
45+78+23=146 [OK]
146+91=237 [OK]
... [repeats all steps]

Alternative method:
(45+12)+(78+89)+(23+34)
+(91+67)=57+167+57+158
... [3rd verification]
\boxed{439}
```

Analysis: Both models arrive at the correct answer, but Phi-4-reasoning uses 17.8× more tokens on redundant verification. The model performs the same calculation multiple times using different groupings without improving accuracy. This exemplifies overthinking where models generate plausible-looking verification text without genuine computational benefit.

Model	Budget	Acc (%)	Inst (%)	O-Score	Tokens	Words	Chars
		Mean ± Std	*Mean ± Std*		*Mean ± Std*	*Mean ± Std*	*Mean ± Std*
Gemini-2.5-Flash							
Gemini-2.5-Flash	disabled	92.0 ± 22.0	95.0 ± 15.0	0.916	389.3 ± 192.4	208.0 ± 99.0	1058.7 ± 521.0
Gemini-2.5-Flash	4915	**93.0** ± 20.0	95.0 ± 15.0	0.927	401.0 ± 195.2	212.3 ± 95.8	1081.4 ± 512.2
Gemini-2.5-Flash	9830	**93.0** ± 20.0	94.0 ± 16.0	0.928	380.5 ± 176.7	203.3 ± 91.0	1033.9 ± 474.7
Gemini-2.5-Flash	14746	**93.0** ± 20.0	95.0 ± 16.0	0.926	389.3 ± 189.1	207.3 ± 97.8	1058.8 ± 525.0
Gemini-2.5-Flash	19661	92.0 ± 22.0	95.0 ± 13.0	0.920	375.7 ± 174.3	200.4 ± 89.5	1020.4 ± 470.2
Gemini-2.5-Flash	24576	92.0 ± 23.0	95.0 ± 14.0	0.918	386.4 ± 189.2	205.9 ± 97.0	1044.8 ± 502.5
Gemini-2.5-Flash	dynamic	92.0 ± 22.0	95.0 ± 15.0	0.919	385.0 ± 187.1	205.0 ± 95.3	1043.4 ± 495.2
Gemini-2.5-Flash-Lite							
Gemini-2.5-Flash-Lite	disabled	83.0 ± 31.0	100.0 ± 2.0	0.830	704.7 ± 362.9	363.6 ± 147.0	1926.0 ± 898.1
Gemini-2.5-Flash-Lite	512	83.0 ± 31.0	100.0 ± 1.0	0.829	718.0 ± 419.2	389.5 ± 229.0	2011.4 ± 1279.9
Gemini-2.5-Flash-Lite	5325	**84.0** ± 32.0	100.0 ± 1.0	0.840	676.6 ± 322.6	365.0 ± 157.2	1859.9 ± 857.8
Gemini-2.5-Flash-Lite	10138	**84.0** ± 32.0	100.0 ± 2.0	0.841	692.8 ± 355.6	363.6 ± 148.8	1883.3 ± 872.2
Gemini-2.5-Flash-Lite	14951	**84.0** ± 32.0	100.0 ± 2.0	0.838	727.5 ± 434.7	376.5 ± 167.7	1970.2 ± 1083.4
Gemini-2.5-Flash-Lite	19764	83.0 ± 31.0	100.0 ± 1.0	0.829	731.7 ± 442.8	381.9 ± 180.1	1949.2 ± 946.5
Gemini-2.5-Flash-Lite	24576	**84.0** ± 31.0	100.0 ± 2.0	0.836	748.0 ± 546.9	390.5 ± 224.2	2035.3 ± 1545.8
Gemini-2.5-Flash-Lite	dynamic	**84.0** ± 32.0	100.0 ± 2.0	0.837	718.0 ± 521.6	384.5 ± 254.4	2007.6 ± 1659.9
Gemini-2.5-Pro							
Gemini-2.5-Pro	128	**90.0** ± 19.0	87.0 ± 22.0	0.904	306.0 ± 122.0	160.5 ± 55.3	846.3 ± 270.0
Gemini-2.5-Pro	6554	82.0 ± 30.0	78.0 ± 31.0	0.827	278.3 ± 140.7	145.9 ± 65.7	768.3 ± 328.3
Gemini-2.5-Pro	12979	71.0 ± 37.0	74.0 ± 37.0	0.722	244.8 ± 161.5	129.1 ± 79.2	691.1 ± 411.8
Gemini-2.5-Pro	19405	89.0 ± 21.0	87.0 ± 22.0	0.893	311.8 ± 128.4	162.8 ± 57.3	861.3 ± 280.2
Gemini-2.5-Pro	25830	89.0 ± 21.0	88.0 ± 19.0	0.906	306.8 ± 124.4	160.9 ± 56.4	849.6 ± 274.6
Gemini-2.5-Pro	32256	88.0 ± 22.0	88.0 ± 21.0	0.892	307.8 ± 128.1	161.1 ± 57.9	849.7 ± 282.9
Gemini-2.5-Pro	dynamic	**90.0** ± 19.0	87.0 ± 21.0	0.903	309.6 ± 128.2	161.8 ± 57.1	853.9 ± 280.9
GPT-5							
GPT-5	minimal	84.0 ± 29.0	100.0 ± 0.0	0.860	158.2 ± 148.9	70.3 ± 54.1	352.4 ± 301.9
GPT-5	low	96.0 ± 16.0	99.0 ± 4.0	0.980	19.3 ± 31.5	7.3 ± 10.8	45.1 ± 57.6
GPT-5	medium	**97.0** ± 15.0	100.0 ± 2.0	0.989	20.5 ± 31.7	7.7 ± 10.7	46.8 ± 57.4
GPT-5	high	**97.0** ± 15.0	100.0 ± 2.0	0.988	24.2 ± 33.2	9.5 ± 11.9	56.4 ± 63.0
GPT-5-Mini							
GPT-5-Mini	minimal	90.0 ± 20.0	98.0 ± 7.0	0.913	180.0 ± 166.0	74.0 ± 62.2	401.6 ± 346.5
GPT-5-Mini	low	94.0 ± 18.0	97.0 ± 11.0	0.958	22.8 ± 31.7	8.9 ± 10.9	53.4 ± 58.3
GPT-5-Mini	medium	96.0 ± 16.0	100.0 ± 0.0	0.977	33.4 ± 35.4	14.4 ± 13.0	81.6 ± 68.0
GPT-5-Mini	high	**97.0** ± 15.0	100.0 ± 0.0	0.983	37.0 ± 34.2	16.2 ± 12.7	90.1 ± 66.3
GPT-5-Nano							
GPT-5-Nano	minimal	52.0 ± 40.0	98.0 ± 6.0	0.571	65.5 ± 94.5	29.5 ± 40.3	159.2 ± 218.7
GPT-5-Nano	low	93.0 ± 19.0	99.0 ± 4.0	0.949	14.5 ± 31.3	4.1 ± 10.5	29.1 ± 56.7
GPT-5-Nano	medium	94.0 ± 17.0	98.0 ± 8.0	0.958	15.8 ± 31.5	4.9 ± 10.5	33.2 ± 56.9
GPT-5-Nano	high	**96.0** ± 15.0	99.0 ± 4.0	0.978	15.6 ± 31.6	4.8 ± 10.6	32.5 ± 57.2
O3							
O3	low	**97.0** ± 15.0	100.0 ± 0.0	0.981	51.6 ± 48.1	25.3 ± 22.9	135.4 ± 111.8
O3	medium	**97.0** ± 15.0	100.0 ± 0.0	0.979	55.4 ± 47.1	27.2 ± 22.8	145.0 ± 111.8
O3	high	**97.0** ± 16.0	100.0 ± 0.0	0.981	55.6 ± 48.3	26.8 ± 22.9	144.1 ± 114.3
O3-Mini							
O3-Mini	low	**93.0** ± 18.0	99.0 ± 3.0	0.936	340.3 ± 368.2	154.7 ± 142.7	782.2 ± 743.4
O3-Mini	medium	**93.0** ± 19.0	99.0 ± 5.0	0.938	357.7 ± 382.8	160.5 ± 142.9	821.2 ± 764.5
O3-Mini	high	**93.0** ± 20.0	98.0 ± 7.0	0.935	367.9 ± 372.9	169.3 ± 140.5	855.0 ± 724.9
O4-Mini							
O4-Mini	low	**95.0** ± 17.0	100.0 ± 0.0	0.963	65.3 ± 45.5	29.6 ± 19.7	163.1 ± 99.2
O4-Mini	medium	**95.0** ± 18.0	100.0 ± 2.0	0.964	66.0 ± 46.1	29.7 ± 19.6	165.7 ± 105.6
O4-Mini	high	**95.0** ± 16.0	100.0 ± 0.0	0.962	69.0 ± 46.2	30.5 ± 20.4	169.6 ± 103.8

Table 9: **Reasoning Budget Model Performance:** Evaluation results across different reasoning budget configurations for Gemini, GPT-5, and O-series models. Within each model family, **bold** indicates best accuracy and underline indicates second-best accuracy. O-Score represents the Overthinking Score calculated as the harmonic mean of accuracy and token efficiency. Detailed reasoning budget analysis appears in Section 5.

Models	Param (B)	Quant	Accuracy	Instr. Follow	O-Score	Tokens	Words
Qwen2.5 (I)	0.5	FP16	21.31	77.57	0.268	432.3	223.2
Qwen2.5 (I)	0.5	8-bit	21.29	76.79	0.268	431.5	223.3
Qwen2.5 (I)	0.5	4-bit	12.77	77.70	0.176	478.0	260.9
Qwen2.5 (I)	1.5	FP16	43.03	85.45	0.470	264.7	134.1
Qwen2.5 (I)	1.5	8-bit	43.67	86.64	0.472	264.3	133.7
Qwen2.5 (I)	1.5	4-bit	39.42	82.97	0.434	292.1	142.3
Qwen2.5 (I)	3	FP16	45.75	92.35	0.463	331.3	176.5
Qwen2.5 (I)	3	8-bit	48.65	91.99	0.497	341.9	181.7
Qwen2.5 (I)	3	4-bit	41.94	90.97	0.438	301.0	158.8
Qwen2.5 (I)	7	FP16	61.36	96.47	0.568	286.9	149.5
Qwen2.5 (I)	7	8-bit	60.61	96.40	0.564	287.5	149.4
Qwen2.5 (I)	7	4-bit	58.03	96.00	0.550	291.9	152.2
Qwen2.5 (I)	14	FP16	63.74	97.83	0.578	260.2	137.1
Qwen2.5 (I)	14	8-bit	63.86	97.89	0.576	261.2	138.1
Qwen2.5 (I)	14	4-bit	60.94	96.69	0.586	240.9	127.5
Qwen2.5 (I)	32	FP16	72.90	99.26	0.643	260.9	139.1
Qwen2.5 (I)	32	8-bit	73.08	99.20	0.645	261.9	139.6
Qwen2.5 (I)	32	4-bit	72.67	99.37	0.640	260.5	139.2
Qwen2.5 (I)	72	FP16	74.87	97.12	0.591	339.2	179.8
Qwen2.5 (I)	72	8-bit	74.05	96.28	0.577	347.1	184.2
Qwen2.5 (I)	72	4-bit	72.74	94.85	0.549	358.3	188.4

Table 10: Impact of GPTQ quantization on Qwen2.5 family performance. FP16 represents full precision baseline. Larger models (32B, 72B) show minimal accuracy degradation even at 4-bit quantization, while smaller models (0.5B) experience substantial performance loss.

Models	Param (B)	Condition	Accuracy	Instr. Follow	O-Score	Tokens	Words
Qwen3 Family: Unconstrained vs Constrained							
Qwen3	0.6	Unconstrained	49.99	83.85	0.484	3162.8	1620.9
Qwen3	0.6	Constrained	27.00	50.41	0.182	762.3	427.9
Qwen3	1.7	Unconstrained	70.24	86.54	0.555	3157.2	1620.7
Qwen3	1.7	Constrained	30.67	42.52	0.189	778.3	435.5
Qwen3	4	Unconstrained	81.90	91.57	0.580	3091.2	1623.1
Qwen3	4	Constrained	31.73	41.30	0.202	785.9	446.4
Qwen3	8	Unconstrained	82.10	91.58	0.615	3027.8	1584.6
Qwen3	8	Constrained	28.35	42.33	0.186	790.9	451.7
Qwen3	14	Unconstrained	86.52	99.27	0.725	3607.6	1941.2
Qwen3	14	Constrained	35.18	49.89	0.199	753.2	426.9
Qwen3	32	Unconstrained	84.13	93.05	0.627	2845.9	1497.5
Qwen3	32	Constrained	30.22	44.56	0.193	778.6	446.8
Phi-4 Reasoning Family: Unconstrained vs Constrained							
Phi-4-reasoning-plus	14	Unconstrained	69.54	88.89	0.288	6780.7	3972.0
Phi-4-reasoning-plus	14	Constrained	44.33	78.88	0.141	1022.2	537.8
Phi-4-reasoning	14	Unconstrained	72.23	96.21	0.352	6066.2	3710.8
Phi-4-reasoning	14	Constrained	53.48	90.21	0.107	1013.5	554.3
Phi-4-mini-reasoning	3.8	Unconstrained	70.16	89.56	0.659	3171.9	1571.7
Phi-4-mini-reasoning	3.8	Constrained	40.52	59.32	0.251	909.8	495.7

Table 11: Performance comparison under unconstrained and constrained (1024 token limit) generation budgets. Reasoning models show catastrophic degradation when token-limited, with Phi-4-reasoning dropping 18.75 accuracy points (26% relative) and Phi-4-reasoning-plus dropping 25.21 points (36% relative).

Case Study 3: Extreme Overthinking in Division

Task: Calculate 846 / 18

Phi-4 Response (47 tokens):

```
846 / 18 = 47

\boxed{47}
```

Phi-4-reasoning Response (1,456 tokens):

```
Long division:
18 into 84? 18x4=72, 18x5=90
So 4 times, remainder 12

Bring down 6: 126
18 into 126? 18x7=126 (exact)
Answer: 47

Verify: 47x18=?
40x18=720, 7x18=126
Total: 846 [OK]

Factorization:
846=2x3x3x47, 18=2x3x3
Therefore 846/18=47 [OK]

Estimation:
846~900, 18~20
900/20=45 (close) [OK]

\boxed{47}
```

Analysis: The reasoning model performs three separate verification methods (multiplication check, prime factorization, estimation) after already arriving at the correct answer through long division. Each verification method is individually valid but collectively wasteful, consuming 31× more tokens without improving the already-correct result.

Case Study 4: Llama-70B Overthinking on Simple Factorial

Task: Calculate 5! (factorial of 5)

Llama-8B Response (124 tokens):

```
5! = 5 x 4 x 3 x 2 x 1
   = 20 x 3 x 2 x 1
   = 60 x 2 x 1
   = 120 x 1
   = 120
\boxed{120}
```

Llama-70B Response (2,891 tokens):

```
5! means 5 factorial...
Let me calculate step by step:
5 x 4 = 20 [OK]
Let me verify: 5x4 = (5x3) + 5 = 15+5 = 20 [OK]

Now 20 x 3 = 60
Verify: 20x3 = 20+20+20 = 60 [OK]
Also: 60 = 5x4x3 [OK]

Continue: 60 x 2 = 120
Verify: 60x2 = 60+60 = 120 [OK]
Check: 120 = 5x4x3x2 [OK]

Finally: 120 x 1 = 120
[... continues with multiple verification
methods, alternative calculations,
prime factorization, and historical
context of factorials for 2500+ tokens]
\boxed{120}
```

Analysis: Larger model (70B) uses 23× more tokens than smaller sibling (8B) with identical final accuracy. The 70B model performs redundant verifications at each step and introduces unnecessary elaboration about factorial concepts. This demonstrates that parameter scaling amplifies overthinking rather than improving efficiency—the smaller model's direct approach is more practical despite having 8.75× fewer parameters.

Case Study 5: Qwen3-32B Constrained Budget Collapse

Task: Find GCD of 48 and 18

Qwen3-32B Unconstrained (4,523 tokens):

```
Using Euclidean algorithm:
48 = 18 x 2 + 12
18 = 12 x 1 + 6
12 = 6 x 2 + 0
Therefore GCD = 6

Verification method 1: Prime factorization
48 = 2^4 x 3
18 = 2 x 3^2
Common: 2 x 3 = 6 [OK]

Verification method 2: List divisors
48: 1,2,3,4,6,8,12,16,24,48
18: 1,2,3,6,9,18
Common divisors: 1,2,3,6
Greatest: 6 [OK]

[... continues with Stein's algorithm,
extended Euclidean algorithm,
and mathematical properties of GCD]
\boxed{6}
```

Qwen3-32B Constrained 512 tokens (WRONG):

```
Find GCD of 48 and 18...
Using Euclidean algorithm:
48 divided by 18...
48 = 18 x 2 + 12
Now 18 divided by 12...
18 = 12 x 1... wait let me recalculate
18/12 = 1 remainder...
Actually 18 = 12 x 1 + 6
So now 12 divided by 6...
12 = 6 x 2 + 0
Therefore... wait, which was the GCD?
The last non-zero remainder...
So that would be... let me check...
I think it's 12? No wait...
[runs out of tokens mid-calculation]
\boxed{12}
```

Final Answer: 12 (INCORRECT - should be 6)

Analysis: The model achieves correct answer with elaborate reasoning in unconstrained mode but **fails completely** when limited to 512 tokens. Rather than adapting to the constraint by being concise, it attempts the same verbose pattern, runs out of tokens mid-verification, and loses track of the calculation, producing an incorrect answer. This reveals trained dependence on long-form generation patterns that cannot adapt to efficiency requirements.

Case Study 6: GPT-4o-mini Efficiency Advantage

Task: Calculate average of [12, 15, 18, 21]

GPT-4o Response (856 tokens):

```
To find average, sum and divide by count.

Sum calculation:
12 + 15 = 27
27 + 18 = 45
45 + 21 = 66

Count: 4 numbers

Average = 66 / 4

Let me verify division:
66 / 4 = 16.5
Check: 16.5 x 4 = 66 [OK]

Alternative verification:
(12+15+18+21)/4
= (27+39)/4
= 66/4 = 16.5 [OK]

Another approach - median check:
Sorted: 12,15,18,21
Median: (15+18)/2 = 16.5
Mean equals median (arithmetic seq) [OK]

[continues with variance calculation
and statistical properties]
\boxed{16.5}
```

GPT-4o-mini Response (98 tokens):

```
Sum: 12+15+18+21 = 66
Count: 4
Average: 66/4 = 16.5
\boxed{16.5}
```

Efficiency Comparison:

- GPT-4o: 856 tokens, Correct
- GPT-4o-mini: 98 tokens, Correct
- Token ratio: 8.7x
- Both achieve 100% accuracy
- Mini variant demonstrates that conciseness does not sacrifice correctness for basic math

Analysis: The "mini" variant achieves identical accuracy with 8.7× fewer tokens by avoiding unnecessary verifications. This case study demonstrates that model optimization can successfully reduce overthinking—GPT-4o-mini was likely fine-tuned or distilled with efficiency objectives that GPT-4o lacks. The smaller model's direct approach is superior for practical deployment despite potentially lower parameter count.

Case Study 7: O3 Model Showing Diminishing Returns

Task: Is 17 a prime number?

O3 Low Budget (412 tokens):

```
Check if 17 is prime.

Divisibility tests:
17/2 = 8.5 (not divisible)
17/3 = 5.67 (not divisible)
17/5 = 3.4 (not divisible)

Only need to check up to sqrt(17) ~ 4.1

Checked: 2,3
17 is not divisible by any number
except 1 and itself.

Therefore 17 is prime.
\boxed{Yes}
```

O3 High Budget (3,847 tokens):

```
Determining if 17 is prime...

Method 1: Trial division
Test all primes up to sqrt(17):
2: 17/2 = 8.5 [not divisible]
3: 17/3 = 5.67 [not divisible]
5: 17/5 = 3.4 [not divisible]
Conclusion: Prime [OK]

Method 2: Sieve of Eratosthenes
[generates sieve up to 20]
Primes found: 2,3,5,7,11,13,17,19
17 is in the list [OK]

Method 3: Primality patterns
17 = 16 + 1 = 2^4 + 1
Check Fermat primality test...
[... continues with Miller-Rabin test,
Wilson's theorem verification,
prime number theorem discussion,
and historical context of prime 17]

All methods confirm: 17 is prime
\boxed{Yes}
```

Analysis: Both budget levels achieve 100% accuracy (correct answer), but high budget uses 9.3× more tokens. The extended reasoning applies multiple primality tests (trial division, sieve, Fermat test, Miller-Rabin) when simple trial division suffices for a two-digit number. This exemplifies diminishing returns: after the first correct determination, additional verification provides zero accuracy gain while consuming substantial computational resources. The model cannot recognize when it has sufficient evidence and should terminate.

Pathological Case 1: Character Repetition (Qwen3-4B)

Task: Find the mode of [5, 5, 5, 7, 7, 9]
Expected Output: Mode is 5 (appears 3 times)

Actual Output (1024 tokens, truncated):

```
Frequency count:
5 appears: 1, 2, 3 times
7 appears: 1, 2 times
9 appears: 1 time

Mode = 5 (most frequent)
nnnnnnnnnnnnnnnnnnnnnn
nnnnnnnnnnnnnnnnnnnnnn
nnnnnnnnnnnnnnnnnnnnnn
... [900+ 'n' chars]
```

Analysis: The model correctly identifies the mode but then generates over 900 consecutive 'n' characters, suggesting a failure in the stopping mechanism. This behavior appears when models reach the end of their planned output but haven't hit token limits, potentially due to poor calibration of end-of-sequence tokens during training.

Pathological Case 2: Random Tokens (Qwen3-0.6B constrained)

Task: Calculate 567 - 234
Expected Output: 333

Actual Output (constrained, 1024 tokens):

```
Subtract:
567-234
=500-200+67-34
=300+33=333

Verify:
567
-234
----
7-4=3, 6-3=3, 5-2=3
Answer: 333

aaabbbcccdddeeefff1234
56789!@#$%^&*()_+-=
[]{}|;:'"<>?~`zzxxcc
vvbbnnmm,.//\\qwerty
... [800+ random chars]
```

Analysis: Despite arriving at the correct answer (333), the constrained model continues generating random alphanumeric characters and symbols to fill the remaining token budget. This suggests models trained on long-form generation struggle to terminate early when problems are solved quickly, particularly under token constraints that mismatch their expected output length distribution.

Pathological Case 3: Self-Contradiction (O3-Mini high budget)

Task: Find max of [12, 45, 23, 89, 34]
Expected Output: 89

Actual Output (2400 tokens, truncated):

```
Scan: 12,45,23,89,34
max=12, then 45, stays 45,
then 89, stays 89
Maximum is 89

Wait, reconsider... is 89
really largest? Maybe 45...
no 89>45... But what if I
missed something?

Sort: 12<23<34<45<89
Yes, 89 is largest.

Verify each:
89>12? Yes. 89>45? Yes.
89>23? Yes. 89>34? Yes.

Check duplicates? None.

Actually, recalculate from
scratch... [1000+ tokens]

\boxed{89}
```

Analysis: With high reasoning budgets, models sometimes enter loops of self-doubt, repeatedly verifying correct answers and introducing unnecessary uncertainty. While the final answer remains correct, the extended reasoning introduces spurious alternatives and "reconsiderations" that don't improve accuracy but waste computation and potentially confuse users.