Do Large Language Models Truly Grasp Addition? A Rule-Focused Diagnostic Using Two-Integer Arithmetic

Anonymous ACL submission

Abstract

Large language models (LLMs) achieve impressive results on advanced mathematics benchmarks but sometimes fail on basic arithmetic tasks, raising the question of whether they have *truly grasped* fundamental arithmetic rules or are merely relying on pattern matching. To unravel this issue, we systematically probe LLMs' understanding of two-integer addition (0 to 2^{64}) by testing three crucial properties: commutativity ($A + B = B + A$), representation invariance via symbolic remapping (e.g., $7 \mapsto$ Y), and consistent accuracy scaling with operand length. Our evaluation of 12 leading LLMs reveals a stark disconnect: while models achieve high numeric accuracy (73.8–99.8%), they systematically fail these diagnostics. Specifically, accuracy plummets to $\leq 7.5\%$ with symbolic inputs, commutativity is violated in up to 20% of cases, and accuracy scaling is non-monotonic. Interventions further expose this pattern-matching reliance: explicitly providing rules degrades performance by 29.49%, while prompting for explanations before answering merely maintains baseline accuracy. These findings demonstrate that current LLMs address elementary addition via pattern matching, not robust rule induction, motivating new diagnostic benchmarks and innovations in model architecture and training to cultivate genuine mathematical reasoning.

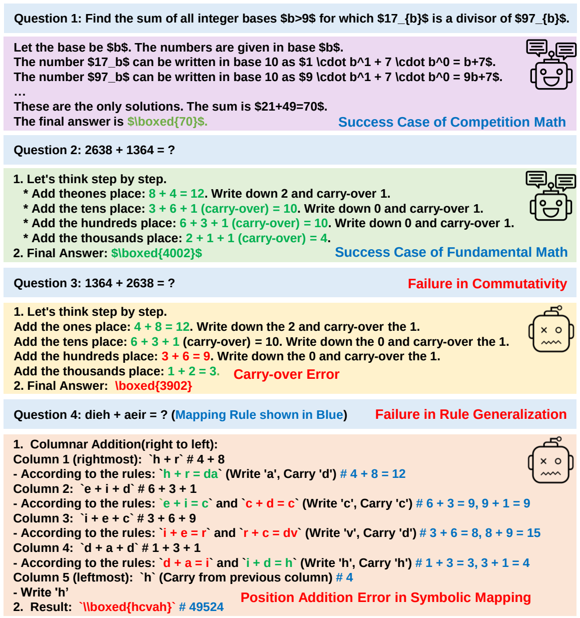

Figure 1: **Illustration of LLM Paradox: LLMs excel at complex math but falter on basic addition, raising the question of whether they grasp rules or merely reproduce patterns.** "True grasp" implies consistent performance and adherence to mathematical properties (e.g., commutativity, representation invariance) under novel conditions. This study probes LLMs' comprehension of elementary two-integer addition (RQ1) and the factors that modulate it (RQ2). Findings suggest that LLMs rely on token-level heuristics rather than rule abstraction.

1 Introduction

Why basic arithmetic still matters. Large Language Models (LLMs) have demonstrated impressive, often graduate-level, performance on sophisticated mathematics benchmarks (OpenAI, 2024; DeepSeek-AI, 2025). This success, however, is frequently contrasted with surprising failures on *elementary* operations when inputs are minimally perturbed, such as reordering operands, substituting digits with symbols, or rephrasing problems (Li et al., 2024; Mirzadeh et al., 2024). This stark contradiction fuels a critical question at the heart of understanding LLM capabilities:

> **Do LLMs truly grasp arithmetic *rules*, or do they primarily reproduce familiar token patterns learned from vast datasets?**

Here, we define *rules* as generalizable algorithms that apply consistently across all valid inputs, and *true grasp* as the ability to maintain performance and adhere to mathematical properties even under novel conditions, reflecting an understanding of the principles themselves.

Diagnostic gap in existing benchmarks. Popular benchmarks such as GSM8K, MATH-500, and Humanity's Last Exam emphasize final-answer accuracy on multi-step word problems (Cobbe et al., 2021; Lightman et al., 2024; Humanity-Team, 2025). Because many sub-steps are unobserved, these benchmarks cannot determine whether success arises from rule learning or from distribution-specific heuristics. Recent robustness probes confirm this concern, but still involve problems whose complexity obscures **which precise rule is violated** (Li et al., 2024; Mirzadeh et al., 2024). A targeted diagnostic is therefore required.

Our diagnostic lens: two-integer addition. We propose elementary two-integer addition as a controlled probe of rule learning (Figure 1). The task isolates a single algorithm with two components, digit-wise addition and carry propagation, and removes linguistic confounds. Any system that genuinely implements this algorithm must satisfy three observable properties: (i) **digit-scaling consistency**, meaning accuracy should decline monotonically with operand length; (ii) **representation invariance**, meaning performance should be stable under any bijective digit-to-symbol remapping; and (iii) **algebraic integrity**, meaning commutativity holds for every operand pair.

Empirical Findings. Our empirical investigation, centered on these properties, reveals that contemporary LLMs predominantly rely on pattern matching rather than exhibiting a robust, rule-based understanding of elementary addition. Key findings highlight significant violations of digit-scaling consistency: numeric accuracy often shows an erratic 'drop-rebound' pattern, such as declining for 4–6 digit operands, then improving for 8–10 digits, instead of the expected monotonic degradation with increasing operand length. LLMs also frequently fail to uphold algebraic integrity; commutativity ($A + B \neq B + A$) is systematically violated in thousands of instances across various models, with some models failing this property in up to 20% of problem pairs. Moreover, performance dramatically collapses under symbolic representation; models achieving over 99% numeric accuracy, like Claude-3.5-Sonnet at 99.81%, experience a performance drop to as low as 7.51% when standard digits are replaced by novel symbols. Interventions such as providing explicit rules via prompting often counterintuitively degrade performance, sometimes by more than 50% relative to zero-shot accuracy. Finally, fine-tuning experiments underscore a persistent tension. Task-specific supervised fine-tuning (SFT) significantly boosts numeric accuracy but typically fails to generalize this improvement to symbolic tasks. In contrast, reinforcement learning (RL) based methods show somewhat better symbolic transfer, although often without matching SFT's peak numeric performance. These outcomes indicate that LLMs' arithmetic competence is often tied to learned surface token patterns, rather than an internalized, abstract grasp of mathematical rules.

Contributions. Our main contributions are:

1. **Diagnostic Methodology:** We introduce a diagnostic methodology centered on two-integer addition, using notation invariance, digit-scaling consistency, and algebraic integrity as key criteria to differentiate genuine rule learning from superficial pattern matching in LLMs.
2. **Empirical Findings:** Through extensive experiments, we demonstrate that current LLMs systematically fail these diagnostic tests, exhibiting significant performance drops with symbolic inputs, erratic scaling, and commutativity violations. Furthermore, interventions like explicit rule provision often impair performance, while fine-tuning highlights a persistent preference for pattern memorization over rule abstraction.
3. **Implications for LLM Evaluation and Development:** These findings reveal a core limitation in LLMs' compositional generalization for elementary arithmetic. This suggests that success on complex benchmarks may mask deficiencies in foundational reasoning, underscoring the need for new evaluation approaches and model architectures to foster genuine mathematical understanding.

2 Related Work

Benchmark progress and limits. Leaderboard-oriented benchmarks have propelled LLMs toward increasingly impressive performance on complex reasoning tasks, ranging from general knowledge assessments to graduate-level mathematics (Hendrycks et al., 2021; Cobbe et al., 2021; MAA, 2024; Humanity-Team, 2025). However, these suites prioritize final answers over the underlying generalizable rules that generate them. Since each problem combines multiple subtasks, high accuracy can be achieved through localized pattern matching, which can evade detection by aggregate metrics. Thus, the fundamental question persists, and our work specifically addresses this uncertainty.

Robustness analyses that reveal surface dependence. Several studies probe models with minimal input perturbations. Replacing digits with unfamiliar symbols, altering numeral formats, or retokenizing inputs consistently reduces accuracy (Mirzadeh et al., 2024; Zhou et al., 2024; Zhong et al., 2024; Zeng et al., 2024). Although specialized embeddings can recover some performance (McLeish et al., 2024), these fixes improve specific surface forms and do not demonstrate rule abstraction. Our notation-remapping experiments extend this line of inquiry by isolating the addition algorithm from every other linguistic cue.

Mechanistic studies of arithmetic circuits. Neuron-level inspections report units that store partial carries, as well as heuristics that fail outside the training range (Qiu et al., 2024; Nikankin et al., 2025). Grokking phenomena illustrate that models can memorize before they generalize, and sometimes never reach full rule induction (Power et al., 2022). Instruction-tuning and in-context exemplars can elicit temporary algebraic behavior, yet systematic transfer remains narrow (Chang and Wu, 2024; Gorceix et al., 2024; Chen et al., 2024; Deng et al., 2024). These findings motivate a diagnostic that tests rule mastery directly rather than inferring it from indirect proxies.

We contribute such a diagnostic by focusing on two-integer addition, a task whose solution requires commutativity and notation invariance but avoids confounds from multi-step language understanding. By evaluating models against these minimal yet stringent criteria, we close the empirical gap left by prior robustness and mechanistic studies and provide a concrete baseline for future architectural and training advances.

3 Methodology

3.1 Background and Motivation

LLMs predominantly employ auto-regressive generation to produce responses. Given a question Q, an LLM samples an answer sequence $A = (T_1, \ldots, T_N)$ from a learned probability distribution of pre-training data:

$$A \sim P(A \mid Q) = \prod_{i=1}^{N} P\left(T_i \mid T_{<i}, T\right), \quad (1)$$

where T in $P\left(T_i \mid T_{<i}, T\right)$ represents the conditioning context. Ideally, this learned distribution $P(A \mid Q)$ should approximate a true underlying distribution $P^*(A \mid Q)$ that reflects genuine comprehension, or *true grasp*, of the principles needed to answer Q. Such genuine comprehension would manifest as consistent behavior; for example, semantically identical questions should elicit responses from similar conditional distributions, and increased question complexity should be reflected in the corresponding distributions. However, empirical evidence indicates that LLMs often deviate from this ideal. High accuracy on complex benchmarks like GSM8K can significantly decrease with minor input perturbations, such as paraphrasing or digit substitution, or on simpler arithmetic tasks (Li et al., 2024; Mirzadeh et al., 2024; Zhou et al., 2024). This discrepancy between the learned $P(A \mid Q)$ and the ideal $P^*(A \mid Q)$ suggests a reliance on surface-level pattern matching rather than internalized, rule-governed reasoning.

While the complexity of word-problem benchmarks obscures precise failure attribution, an ideal diagnostic task should isolate a single, verifiable algorithm and use an input domain too vast for memorization. Two-integer addition, a classic microbenchmark for algorithmic generalization (Saxton et al., 2019), meets these criteria.

Our methodology thus rigorously probes LLMs' grasp of elementary addition, addressing:

RQ1: Do LLMs satisfy key markers of rule-based addition: notation invariance, digit-scaling consistency, and algebraic integrity?

RQ2: Can prompt-level or parameter-level interventions bridge the gap between pattern recall and rule induction?

To answer these questions, we construct a dataset of two-integer addition problems and evaluate LLMs against three diagnostic properties: digit-scaling consistency, representation invariance, and algebraic integrity. We also explore the effects of explicit rule provision and prompting strategies on model performance in following sections.

3.2 Diagnostic Task: Two-Integer Addition

Two-integer addition is our diagnostic task. It is the simplest complete arithmetic algorithm (core components: per-digit addition, carry propagation), requires generalization beyond memorization, and permits unambiguous verification. From these characteristics, **we derive three fundamental properties that any model claiming to have truly**

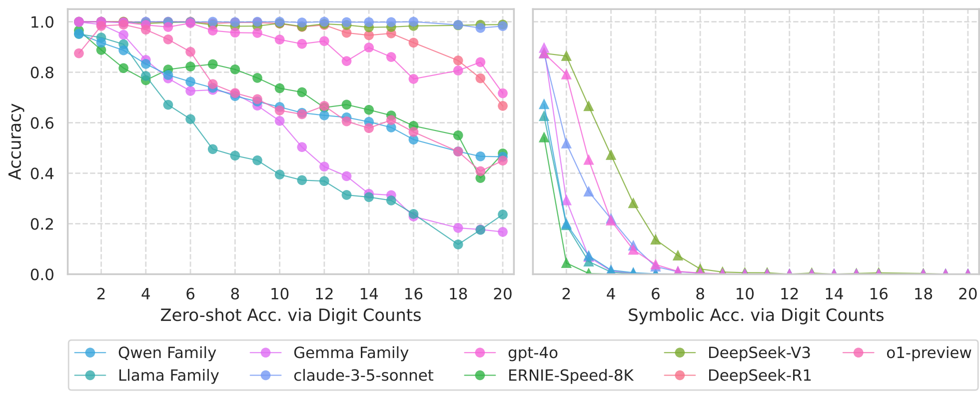

Figure 2: **Performance Degradation Patterns in Zero-shot vs. Symbolic Addition.** While LLMs achieve high accuracy on standard numerical addition (left), their non-monotonic performance curve suggests brittle pattern matching rather than true algorithmic reasoning. In contrast, symbolic addition tests (right) reveal systematic degradation with increasing digit count. This stark contrast between numerical and symbolic performance suggests LLMs rely heavily on memorized patterns rather than learned arithmetic principles.

learned the addition algorithm must consistently demonstrate:

1. **Digit-scaling consistency:** Accuracy should not increase (ideally, it should monotonically decrease) with increasing operand length. This reflects the cumulative error potential inherent in iterative algorithms such as carry propagation. Any deviation from this non-increasing pattern indicates a reliance on length-specific heuristics instead of a general, scalable rule.

2. **Algebraic integrity:** Fundamental algebraic properties, such as commutativity ($A + B = B + A$), must be consistently upheld. A model with a genuine grasp should yield $P(S \mid \text{Query}(A+B)) \approx P(S \mid \text{Query}(B+A))$. Systematic violations of such properties provide direct evidence against true rule grasp.

3. **Representation invariance:** Performance must remain robust when standard digits are bijectively mapped to novel, arbitrary symbols. A significant degradation in performance under such mapping implies that the learned distribution $P(A \mid Q)$ is memorized to surface token patterns, rather than grasping rule of $P^*(A \mid Q)$.

These three properties serve as direct litmus tests for distinguishing genuine rule induction from superficial pattern matching.

3.3 Dataset Construction

To systematically evaluate LLMs against our diagnostic properties, we construct a comprehensive dataset of 100,000 unique two-integer addition problems. The operands A and B are sampled from the range $[0, 2^{64} - 1)$. We structure the dataset generation in three phases to ensure thorough coverage and facilitate targeted analyses:

1. **Phase 1 (Baseline):** All two-digit addition pairs (0-99) for fundamental assessment.
2. **Phase 2 (Digit Scaling):** Uniform sampling of same-length operand pairs (3-20 digits), enabling assessment of length-dependent performance.
3. **Phase 3 (Large Numbers):** Additional samples from 2^{49} to 2^{64}, testing robustness.

To directly assess our diagnostic properties, the dataset incorporates specific structural features. Algebraic integrity, particularly commutativity, is systematically evaluated by including the commuted counterpart (B,A) for every generated ordered pair (A,B). For testing representation invariance, we define ten independent bijective digit-to-symbol mappings (e.g., $0 \mapsto \text{U}, \ldots, 9 \mapsto \text{C}$) and apply them to a designated subset of numerical problems to create corresponding symbolic variants. The dataset is subsequently split into training (80%), validation (10%), and test (10%) portions. To ensure efficiency, evaluations of proprietary, reasoning, and fine-tuned models are restricted to this test set. This comprehensive dataset design facilitates a thorough assessment of both rule-learning criteria and intervention effects through systematic property verification.

4 Experiments

To empirically test our central hypothesis, that LLMs rely on pattern matching over rule induction for elementary addition, and to validate our diagnostic methodology (Section 3), we conducted experiments addressing the two RQs:

Task Type	zero-shot		symbolic	zero-shot		symbolic
T	0.0	0.7	0.7	0.0	0.7	0.7
Violation Threshold #	5	5	5	10	10	10
Llama3-8B-it	11918	1678	41	4783	1	-
Llama3-70B-it	-	5402	506	-	432	20
Llama3.1-8B-it	13324	3232	49	4546	6	-
Llama3.1-70B-it	-	4546	602	-	253	22
Llama3.3-70B-it	-	5086	1122	-	1771	81
Qwen2.5-7B-it	7442	3961	302	3402	151	29
Qwen2.5-72B-it	-	812	745	-	25	10

Table 1: **Commutativity Violations Statistics.** Each entry reports the number of (A,B) pairs, out of 50,000, where the model correctly computes $A+B$ but fails on $B+A$; lower counts indicate better performance. Columns correspond to different decoding temperatures T and minimum thresholds (5 or 10 identical successes out of 10 samples in total).

- **RQ1: Do LLMs satisfy key markers of rule-based addition?** To answer this, we evaluate a diverse set of LLMs against the three core diagnostic properties derived from our methodology: digit-scaling consistency, algebraic integrity (specifically commutativity), and notation invariance.
- **RQ2: Can prompt-level or parameter-level interventions bridge the gap between pattern recall and rule induction?** Here, we probe how LLM performance on addition is modulated by explicit rule provision, self-explanation prompting, and task-specific fine-tuning.

We evaluated a diverse range of contemporary LLMs, including open-source families (e.g., Llama, Qwen) and proprietary LLMs (e.g., GPT-4, Gemini series), with full details in Appendix A.2. These experiments provide empirical evidence for our claims about LLM arithmetic capabilities.

4.1 RQ1: Do LLMs Truly Grasp Addition?

To determine if LLMs have truly grasp the addition rules, as opposed to merely mimicking patterns, we evaluated their performance against the three properties. Failure to satisfy these properties would serve as strong evidence of a superficial understanding, reliant on surface-level heuristics rather than genuine rule learning. We present the empirical findings for each diagnostic in turn.

Violation of Digit-Scaling Consistency. Our first diagnostic, digit-scaling consistency, posits that accuracy should not increase with operand length for a system that has internalized an iterative algorithm like addition. However, LLMs frequently violate this principle. As shown in Figure 2 (left panel), many models exhibit a non-monotonic 'drop-rebound' accuracy curve on numeric inputs: performance initially declines for 4–6 digit operands, then unexpectedly improves for 8–10 digits, before declining again. This erratic scaling suggests that the learned distribution $P(A \mid Q)$ for numeric addition deviates from the ideal $P^*(A \mid Q)$ defined in Section 3.1, indicating reliance on length-specific heuristics or memorized fragments rather than a general, scalable rule. In contrast, when inputs are symbolic (Figure 2, right panel), accuracy, while substantially lower, tends to decline more monotonically with increasing digit count. This latter pattern, paradoxically more aligned with algorithmic processing under stress, reinforces the interpretation that high performance on standard numeric inputs reflects pattern matching rather than the robust rule application outlined by our criteria in Section 3.2.

Failure to Uphold Algebraic Integrity. We assessed algebraic integrity by testing adherence to commutativity, a fundamental property of addition. A violation occurred if a model correctly computed $A + B$ but not $B + A$ (or vice versa) in at least 5 (or all 10) of 10 stochastic decodes per problem, at temperatures $T = 0.0$ and $T = 0.7$. The results (Table 1) reveal frequent and systematic commutativity failures. For instance, some 7-8B Llama and Qwen models processing numerical inputs violated commutativity in approximately 20% of 50,000 problem pairs (when failing $\geq 5/10$ decodes) and in 8.48% (4,243 pairs) when failing all 10 decodes. These inconsistencies persisted even at $T = 0.7$, where 0.104% of pairs (52 instances) showed such persistent violations across all decodes. Conversely, on symbolic tasks, Qwen2.5-7B showed such consistent (10/10) violations in only 0.058% of pairs (29 instances), while Llama models exhibited none. Moreover, the observation that larger Llama models were sometimes more prone to these violations challenges the notion that increased model scale inherently confers deeper arithmetic understanding. Such widespread and systematic failures strongly suggest inherent deficiencies in the models' understanding of the addition algorithm, rather than random noise.

Breakdown under Tests of Notation Invariance. Our third diagnostic, notation invariance, assesses if LLM performance on addition remains stable when standard digits are bijectively mapped to novel symbols, a property expected if the abstract addition algorithm is truly internalized. As detailed in Table 2, all tested models dramatically failed this test. Even those with near-perfect accuracy on numeric inputs (e.g., 99.81% for Claude-3.5-Sonnet)

Task Type	Overall Acc.			Position Add Acc.			Carry-over Acc.		
	ZS	S	Δ	ZS	S	Δ	ZS	S	Δ
Gemini2.0-pro-exp	94.88	14.21	-80.67	69.52	4.19	-65.33	77.36	7.07	-70.29
Claude-3.5-Sonnet	99.81	7.51	-92.30	81.78	3.19	-78.59	90.28	6.92	-83.36
GPT-4o	93.39	9.59	-83.80	76.12	3.79	-72.33	79.55	6.73	-72.82
DeepSeek-V3	98.92	16.14	-82.78	78.55	11.98	-66.57	81.14	15.23	-65.91
Gemma2-9b-it	66.34	1.45	-64.89	58.52	0.34	-58.18	60.44	0.44	-59.99
Gemma2-27b-it	83.65	2.62	-81.03	74.77	0.91	-73.85	76.68	0.91	-75.77
Llama3.1-8B-it	43.34	0.57	-42.76	20.38	0.10	-20.27	21.96	0.25	-21.72
Llama3.1-70B-it	72.58	2.51	-70.07	60.13	0.52	-59.61	61.05	1.33	-59.71
Qwen2.5-7B-it	83.00	0.58	-82.41	71.39	0.11	-71.28	74.49	0.13	-74.37
Qwen2.5-72B-it	96.07	5.97	-90.10	88.19	2.09	-86.10	89.78	4.12	-85.67

Table 2: **Accuracy on elementary two-integer addition. ZS** = zero-shot numeric form; **S** = Symbolic form (bijective digit-to-symbol mapping);

Task	Task Type	Llama3.1-8B-it	Qwen2.5-7B-it
Add	symbolic	0.64	0.58
	zero-shot	43.43	83.03
Multi	symbolic	0.01	0.04
	zero-shot	9.92	17.29
Sub	symbolic	0.02	0.01
	zero-shot	18.39	43.88

Table 3: **Performance on Other Arithmetic Operations.** Building on observed difficulties with addition, we evaluated subtraction and multiplication. LLMs performed poorly on symbolic representations of these operations, with low zero-shot accuracy. This suggests their struggles extend to these more complex operations and their symbolic forms.

saw performance collapse on symbolic equivalents, to as low as 7.51%. This failure extended to fundamental components like positional addition and carry-over sub-tasks once familiar digit patterns were absent. Such pronounced inability to generalize to novel symbols strongly indicates that current LLMs rely on recognizing and reproducing patterns tied to standard decimal representations, rather than having learned an abstract, symbol-agnostic addition rule.

Collectively, the evidence from these three diagnostic tests converges on a clear and consistent conclusion: despite often achieving high accuracy on standard numeric addition problems, contemporary LLMs do not demonstrate a robust, rule-based understanding of this elementary operation. Their competence appears tightly coupled to familiar surface token patterns and specific operand lengths, and it degrades systematically when these patterns are disrupted or when fundamental algebraic properties are rigorously tested. This pattern of behavior strongly indicates a primary reliance on pattern matching rather than genuine rule induction for performing elementary addition.

Having established these fundamental deficiencies in LLMs' grasp of basic addition, we next investigate factors that might modulate this understanding in Section 4.2.

4.2 RQ2: What factors modulate grasping?

The preceding analysis (RQ1) demonstrated LLMs' significant deficiencies in internalizing elementary addition rules. To further understand the nature of these limitations and explore potential avenues for improvement, RQ2 investigates factors that might modulate LLMs' ability to grasp these rules. We examine two primary categories of interventions: (1) prompt-level strategies, including the provision of explicit rules and the use of self-explanation prompts, and (2) parameter-level modifications through task-specific fine-tuning.

4.2.1 Explicit Rule Provision

Building on RQ1's finding that LLMs struggle with genuine arithmetic understanding, this subsection investigates whether explicit rule provision can enhance their performance. We evaluated LLMs under several prompt-level interventions: few-shot prompting with definitions of addition principles and examples of varying digit lengths (denoted `Few-Shot`, `Few-Shot-2`, and `Few-Shot-3`), and an `Explain-and-Do` strategy, where models first articulate their problem-solving approach. Results are presented in Table 4 and Figure 3.

Our investigation reveals a counterintuitive finding: providing LLMs with abstract addition rules consistently degraded performance compared to zero-shot settings. This suggests LLMs favor memorizing token patterns over abstracting principles. When faced with human-articulated rules (e.g., "carry the 1"), models struggle to operationalize them, defaulting to pre-trained pattern-matching. This preference explains performance disparities between numerical and symbolic tasks and observed commutativity violations. In contrast, the `Explain-and-Do` strategy—prompting models to first articulate their reasoning—generally maintained performance near zero-shot levels. These findings indicate current LLMs are predominantly optimized for pattern recognition, not abstract rule learning, highlighting a divergence from human mathematical cognition.

Architectural differences among LLM families influenced their responses to interventions. Llama models, despite sometimes lower zero-shot numerical accuracy than Qwen or Gemma (RQ1), adapted better to explicit rules, especially with the `Explain-and-Do` strategy; some variants (e.g., Llama3.1, Llama3.2) even surpassed zero-shot baselines, showing modest compositional generalization gains (Table 4). Conversely, Qwen models,

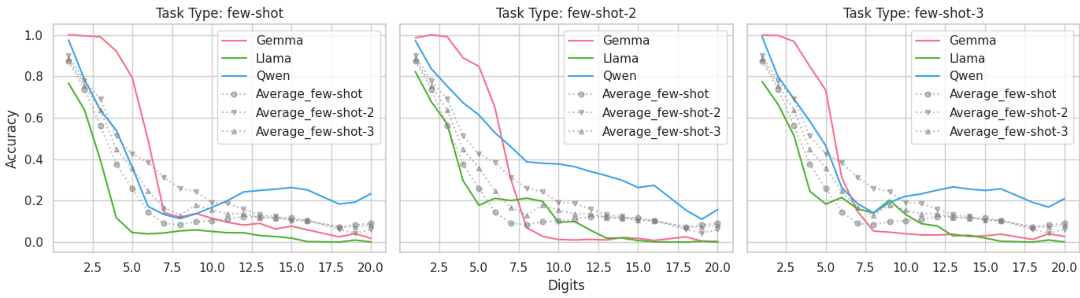

Figure 3: **Few-Shot Performance with Explicit Rule Provision.** Explicit rule provision leads to a significant drop in performance compared to zero-shot, contradicting the expected improvement.

Task Type	Carry-over Acc.						Position Add Acc.					
Models	ZS	S	FS	FS-2	FS-3	E	ZS	S	FS	FS-2	FS-3	E
Llama3-8b-it	15.89	0.20	8.42	15.38	16.68	13.54	16.25	0.07	7.15	15.00	14.34	12.32
Llama3.1-8b-it	21.96	0.25	8.84	15.33	12.46	24.80	20.38	0.10	7.92	12.91	10.14	23.61
Llama3.2-11b-it	17.35	0.26	9.04	19.70	13.97	27.47	16.60	0.12	8.29	18.92	12.57	27.13
Qwen1.5-7b-it	47.44	0.09	3.09	6.40	5.36	7.51	46.78	0.05	2.66	5.98	4.62	8.00
Qwen2-7b-it	62.94	0.06	28.36	57.22	32.35	70.83	60.03	0.05	23.65	48.34	28.25	68.80
Qwen2.5-7b-it	74.49	0.13	38.28	55.08	41.54	72.09	71.39	0.11	33.16	48.12	36.11	71.53

Table 4: **Impact of Different Knowledge Intervention Strategies.** Contrary to expectations, providing explicit rules (few-shot conditions) significantly reduces performance compared to zero-shot baseline, e.g. "Qwen2.5-7b-it" drop 29.49%. However, when models explain their reasoning before computation (explain-and-do), performance remains comparable to zero-shot levels. *ZS = Zero-Shot*, *FS = Few-Shot*, *E = Explain-and-Do*.

despite stronger initial zero-shot performance, degraded more with explicit rules and showed less compositional generalization. These variations suggest fundamental architectural differences in knowledge encoding and access, affecting prompt responsiveness beyond mere computational capacity. While model upgrades generally improve performance, core pattern recognition tendencies persist. Overall, this indicates LLMs primarily rely on pattern matching, highlighting a persistent limitation in abstract, rule-based reasoning.

4.2.2 Rule Internalization

We then investigated if parameter-level modifications via fine-tuning could improve LLMs' internalization of arithmetic rules, moving beyond mere pattern matching. We explored various fine-tuning strategies: SFT, RL with Direct Preference Optimization (DPO) (Rafailov et al., 2023), and a hybrid RPO (SFT+DPO) (Pang et al., 2024). Model performance was assessed on both numerical and symbolic addition post-fine-tuning, and benchmarked against specialized mathematical reasoning models like Eurus2 (Cui et al., 2025), OpenAI o1 (OpenAI, 2024), DeepSeek R1 (DeepSeek-AI, 2025), and their distilled counterparts. For RL, training data comprised model responses, with correct and incorrect answers from our dataset serving as positive and negative examples, respectively (details in Appendix A.3).

Fine-tuning experiments (Table 5) revealed clear trade-offs. Task-specific SFT boosted performance on in-domain numerical addition but failed to generalize to symbolic one, indicating that SFT primarily reinforces pattern matching tied to data. Conversely, RL-based methods (DPO and RPO) achieved better generalization to symbolic inputs, albeit with lower absolute accuracy on the fine-tuned numerical task. Notably, the RPO still struggled with symbolic transfer, suggesting SFT's propensity for pattern matching can overshadow RL's generalization benefits. These findings imply that standard fine-tuning, particularly SFT, optimizes for surface-level pattern recognition over the abstraction of underlying arithmetic principles.

Supporting this, models fine-tuned on general-domain reasoning objectives (e.g., DS-R1-Distill) demonstrated more robust generalization to symbolic tasks. This improved transfer is likely due to training objectives that promote extended reasoning, highlighting the training paradigm's crucial role in fostering generalizable mathematical skills. In contrast, domain-specific models like Eurus2-SFT and Eurus2-PRIME, despite excelling at complex numerical tasks within their domain, showed limited transfer to symbolic addition. However,

Task Type Models	Fine-Tuning Type	Dataset Domain	Overall Acc.			Position Add Acc.			Carry-over Acc.			Map Acc.
			ZS	S	Δ	ZS	S	Δ	ZS	S	Δ	S
Qwen2.5-7B-it	-	-	83.00	0.58	-82.41	71.39	0.11	-71.28	74.49	0.13	-74.37	0.57
Eurus2-7B-SFT	SFT	Domain Specific	83.21	0.42	-82.79	81.21	3.19	-78.02	82.28	6.87	-75.41	-
Eurus2-7B-PRIME	RL(PRM)	Domain Specific	94.11	1.03	-93.08	91.59	3.10	-88.49	92.51	3.11	-89.40	-
DS-R1-Distill-Qwen-7B	RL(Reasoning)	General	74.76	6.88	-67.88	65.38	33.41	-31.97	64.27	31.52	-32.75	-
Qwen2.5-7B-it	SFT	Task Specific (Numerical)	97.17	0.00	-97.17	87.91	0.25	-87.66	89.51	1.26	-88.25	8.21
Qwen2.5-7B-it	RL(DPO)	Task Specific (Numerical)	95.32	0.37	-94.95	86.23	1.17	-85.06	87.75	2.35	-85.40	2.25
Qwen2.5-7B-it	RL(SFT+DPO)	Task Specific (Numerical)	96.95	0.28	-96.67	84.48	0.29	-84.19	85.52	0.61	-84.91	0.10
Qwen2.5-7B-it	SFT	Task Specific (Symbolic)	0.00	30.66	+30.66	3.40	3.89	+0.49	6.71	6.98	+0.27	23.49
Qwen2.5-7B-it	RL(DPO)	Task Specific (Symbolic)	50.73	24.10	-26.63	47.71	3.48	-44.23	48.40	6.37	-42.03	19.84
Qwen2.5-7B-it	RL(SFT+DPO)	Task Specific (Symbolic)	12.32	2.85	-9.47	9.31	0.58	-8.73	9.70	1.13	-8.57	2.00

Table 5: **Impact of Fine-Tuning Approaches on Arithmetic Capabilities.** Different fine-tuning strategies and dataset domains yield distinct trade-offs between performance and generalization. While SFT achieves highest numerical accuracy, it shows minimal transfer to symbolic tasks. RL-based approaches demonstrate better generalization but lower absolute performance. Task-specific training on numerical data excels within-domain but fails to transfer, whereas general-domain training (e.g., DS-R1-Distill) enables broader generalization through its diverse training objectives, suggesting the importance of training paradigm design in developing robust mathematical capabilities.

Task Type	Position Add Acc.			Carry-over Acc.		
	ZS	S	Δ	ZS	S	Δ
Gemini2.0-pro-exp	69.52	4.19	-65.33	77.36	7.07	-70.29
Gemini2.5-pro-exp (thinking)	88.97	19.80	-69.17	88.49	24.56	-63.93
Llama3.3-70b-it	73.82	0.77	-73.05	75.00	2.43	-72.57
DS-R1-Distill-Llama-70B	68.91	42.94	-25.97	68.56	40.75	-27.81
Llama3.1-8b-it	20.38	0.10	-20.27	21.96	0.25	-21.72
DS-R1-Distill-Llama-8B	45.54	39.55	-5.99	44.16	35.09	-9.07

Table 6: **LLMs' Understanding of Addition Principles.** Models achieve high accuracy(%) on standard numerical tasks (zero-shot) but show severe degradation when tested on symbolic representations, both for carry operations and digit addition. This stark contrast suggests that models only grasp principles in numerical form and fail to generalize to abstract representations.

Eurus2-PRIME generalized better than Eurus2-SFT. This suggests RL-based signals can aid in abstracting principles, though balancing specialization with generalization remains challenging.

Specialized reasoning models (Table 6) offered further insights. These models typically showed less performance degradation on symbolic addition compared to standard LLMs, suggesting that training on prolonged or complex reasoning tasks can foster better abstraction of arithmetic principles. Yet, this improved abstraction may entail a trade-off: some reasoning-focused architectures sacrificed accuracy on elementary computations (Figure 2), potentially by "over-thinking" simple problems despite excelling at complex ones. This pattern underscores how architectural design and training objectives critically shape the balance between foundational computational skills and higher-order reasoning.

5 Conclusion

Our empirical results of two-integer addition task reveal that **LLMs fail to grasp elementary addition rules, still relying instead on surface-level pattern matching.** This conclusion is evidenced by: (1) a collapse in accuracy (e.g., from $\geq 99\%$ to $\leq 7.5\%$) when standard digits are replaced with novel symbols, demonstrating a lack of notation invariance; (2) non-monotonic accuracy scaling with operand length, suggesting specific memorization over consistent carry-propagation; and (3) systematic commutativity violations, which contradict genuine rule grasp. These findings collectively indicate that LLMs' success on complex math benchmarks may mask a superficial understanding of basic rules.

Interventions further highlight these deficits: providing formal rules from human knowledge paradoxically degrades performance (by up to 81.2%), while prompting models to "Explain-and-Answer" merely preserves baseline scores. Task-specific SFT boosts numeric accuracy but fails to generalize to symbolic tasks; conversely, RL shows better symbolic transfer but at the cost of lower absolute accuracy. This suggests a fundamental misalignment between human-like abstract rule learning and the pattern-matching heuristics LLMs develop during pre-training.

The implications are significant: current benchmarks, rewarding final answers over rule fidelity, risk inflating perceived LLM competence. Future evaluations must test notation invariance, scaling consistency, and algebraic integrity. Model design should explore explicit symbolic manipulation or execution-grounded reasoning. Bridging the pattern recall-rule abstraction gap is crucial for genuine mathematical understanding in LLMs.

Ethical Considerations

While our research primarily focuses on the mathematical reasoning capabilities of LLMs, which not directly involve ethical considerations. However, the implications of our findings extend to broader ethical concerns in AI deployment. We highlight following key areas:

Why arithmetic robustness matters. Elementary addition underpins many downstream computations. A model that answers graduate-level problems yet violates commutativity can silently corrupt applications that rely on implicit arithmetic, including dose calculation, portfolio rebalancing, and automated bidding. This gap between perceived and actual competence creates a direct safety hazard.

Inflated competence metrics. Public leaderboards optimise for final-answer accuracy rather than rule fidelity. Our results show that such metrics can conceal thousands of systematic arithmetic errors. Deploying models on the basis of these scores may therefore foster unwarranted confidence and expose users to financial or physical harm.

Recommendations for high-stakes deployment. Before adoption in safety-critical settings, developers should (i) report notation-invariance and algebraic-integrity scores alongside aggregate benchmarks, (ii) document failure modes such as the symbol-mapping collapse identified here, and (iii) install run-time monitors that flag out-of-distribution numeric inputs. These measures align claimed capability with real-world reliability.

Toward stronger evaluation standards. The field needs public, reproducible suites that test formal properties directly, not just end-to-end accuracy. Without such standards, the gap between apparent and actual mathematical competence will widen and public trust in AI will erode.

With these considerations in mind, we would highlight the importance and significance these findings have for the future of AI systems. As LLMs are increasingly integrated into various domains, ensuring their reliability and robustness in fundamental tasks like arithmetic is crucial for safe and effective deployment.

Limitations

Scope of mathematical operations. Our study targets two-integer addition because it offers a clean probe of rule learning. Preliminary experiments suggest similar failures in subtraction, multiplication, and symbolic logic, but verifying those trends remains future work.

Range of intervention techniques. We evaluate prompt engineering, SFT, and preference-based RL. Alternative strategies—such as modular arithmetic heads, execution-augmented decoding, or neuro-symbolic hybrids—may yield different generalisation patterns that we have not explored.

External validity of the synthetic dataset. The symbol-mapping protocol strips away contextual cues that may aid reasoning. In real documents, numeric reasoning is embedded in richer text, so model behaviour could differ. Future studies should embed the same invariance checks in realistic narratives such as medical charts or financial statements.

Sampling constraints. API costs limited us to fewer than ten stochastic decodes for some proprietary models. Although the observed failure margins are large, denser sampling would narrow confidence intervals.

Mechanistic understanding. We observe strong evidence of pattern matching rather than rule induction, yet the circuit-level mechanisms remain unidentified. Tracing these mechanisms and designing architectures that promote rule abstraction are important directions for future research.

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, and 8 others. 2024. Phi-4 technical report. *Preprint*, arXiv:2412.08905.

Fu-Chieh Chang and Pei-Yuan Wu. 2024. Unraveling arithmetic in large language models: The role of algebraic structures. *Preprint*, arXiv:2411.16260.

Junhao Chen, Shengding Hu, Zhiyuan Liu, and Maosong Sun. 2024. States hidden in hidden states: Llms emerge discrete state representations implicitly. *Preprint*, arXiv:2407.11421.

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.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, and 4 others. 2025. Process reinforcement through implicit rewards. *Preprint*, arXiv:2502.01456.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Chunyuan Deng, Zhiqi Li, Roy Xie, Ruidi Chang, and Hanjie Chen. 2024. Language models are symbolic learners in arithmetic. *Preprint*, arXiv:2410.15580.

Antoine Gorceix, Bastien Le Chenadec, Ahmad Rammal, Nelson Vadori, and Manuela Veloso. 2024. Learning mathematical rules with large language models. *Preprint*, arXiv:2410.16973.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Humanity-Team. 2025. Humanity's last exam. *Preprint*, arXiv:2501.14249.

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.

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

MAA. 2024. American invitational mathematics examination 2024. `https://huggingface.co/datasets/Maxwell-Jia/AIME_2024`.

Sean Michael McLeish, Arpit Bansal, Alex Stein, Neel Jain, John Kirchenbauer, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, Jonas Geiping, Avi Schwarzschild, and Tom Goldstein. 2024. Transformers can do arithmetic with the right embeddings. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2024. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *Preprint*, arXiv:2410.05229.

Yaniv Nikankin, Anja Reusch, Aaron Mueller, and Yonatan Belinkov. 2025. Arithmetic without algorithms: Language models solve math with a bag of heuristics. In *The Thirteenth International Conference on Learning Representations*.

OpenAI. 2024. Openai o1 system card. `https://cdn.openai.com/o1-system-card-20241205.pdf`.

Richard Yuanzhe Pang, Weizhe Yuan, He He, Kyunghyun Cho, Sainbayar Sukhbaatar, and Jason Weston. 2024. Iterative reasoning preference optimization. In *Advances in Neural Information Processing Systems*, volume 37, pages 116617–116637. Curran Associates, Inc.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. 2022. Grokking: Generalization beyond overfitting on small algorithmic datasets. *Preprint*, arXiv:2201.02177.

Luyu Qiu, Jianing Li, Chi Su, Chen Jason Zhang, and Lei Chen. 2024. Dissecting multiplication in transformers: Insights into llms. *Preprint*, arXiv:2407.15360.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pages 53728–53741. Curran Associates, Inc.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. In *International Conference on Learning Representations*.

Zhongshen Zeng, Pengguang Chen, Shu Liu, Haiyun Jiang, and Jiaya Jia. 2024. Mr-gsm8k: A meta-reasoning benchmark for large language model evaluation. *Preprint*, arXiv:2312.17080.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. 2024. Generative verifiers: Reward modeling as next-token prediction. *Preprint*, arXiv:2408.15240.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. 2024. Sglang: Efficient execution of structured language model programs. *Preprint*, arXiv:2312.07104.

Qihuang Zhong, Kang Wang, Ziyang Xu, Juhua Liu, Liang Ding, and Bo Du. 2024. Achieving >97% on GSM8k: Deeply understanding the problems makes LLMs better solvers for math word problems. *Preprint*, arXiv:2404.14963.

Zhejian Zhou, JIayu Wang, Dahua Lin, and Kai Chen. 2024. Scaling behavior for large language models regarding numeral systems: An example using pythia. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3806–3820, Miami, Florida, USA. Association for Computational Linguistics.

A Appendix

A.1 AI Use Statement

This research utilized AI assistance for code debugging and grammatical refinement. All experimental designs, analyses, results, and conclusions were developed independently by the authors without generative AI input. We employed AI tools solely for technical implementation support and language polishing to ensure clear communication of our findings.

A.2 Experimental Setup

Our evaluation framework utilized the SGLang platform through the official Docker container `lmsysorg/sglang` (Zheng et al., 2024). For statistical robustness, most models underwent 10 repeated evaluations per test example using a temperature setting of 0.7 across the full dataset. Due to computational and budget constraints, select models including `GPT4-o`, `Claude-3.5-Sonnet`, `QwQ-32B-Preview`, `Deepseek-R1` and its variants were evaluated once on the test split only.

For assessing `Position Addition` and `Carry-over` accuracy, we used `Phi-4` (Abdin et al., 2024) as an independent generative evaluator following Zhang et al. (2024). Solutions were evaluated by feeding them to the evaluator to determine carry-over and position addition correctness, using the first token as the prediction.

We conducted comprehensive evaluations across all model variants in both zero-shot and symbolic settings, with complete results presented in Table 7.

A.3 Fine-Tuning Configuration

Our investigation employed three fine-tuning approaches: standard DPO, RPO (combining DPO with SFT), and pure SFT. Each approach shared core configuration elements while varying key method-specific parameters.

Base Configuration. The base configuration utilized a batch size of 1 sample per device with 4 gradient accumulation steps (effective batch size of 4). Training ran for 1 epoch using cosine learning rate scheduling with 10% warmup steps. We implemented BF16 mixed precision and non-reentrant gradient checkpointing, evaluating on a 1% validation set every 500 steps. Flash Attention 2 optimized computation efficiency.

Distributed Training. Training leveraged DeepSpeed ZeRO-3 with 8 processes per machine. The implementation included CPU optimizer state offloading, gradient clipping at 1.0, 16-bit parameter saving, and static process coordination through DeepSpeed's rendezvous mechanism.

Method-specific Parameters.

- Standard DPO: Learning rate 5.0×10^{-6}, $\beta = 0.0$, sigmoid loss function
- RPO: DPO settings with $\beta = 1.0$ for integrated preference modeling and SFT
- SFT: Learning rate 1.0×10^{-4} for supervised training

All approaches utilized full-parameter fine-tuning through DeepSpeed ZeRO-3. For preference learning (DPO/RPO), we initialized reference models from SFT checkpoints with preference loss weight (λ_{ftx}) set to 1.0.

Infrastructure. Training infrastructure consisted of 4 NVIDIA A100 GPUs (80GB each), with complete fine-tuning requiring approximately 15 hours per run.

Prompt Template for Zero-Shot Setting

Context:
You are a helpful AI assistant.

Instruction:
Present your solution in the following format:
1. Let's think step by step.
2. Final Answer: Express using LaTeX notation `\boxed{answer}`

Question:
`%s + %s = \boxed{?}`

Figure 4: **Zero-Shot Setting Prompt Template.** Example prompt template for zero-shot addition tasks, providing context, instructions, and question format for LLMs.

Prompt Template for Few-Shot Setting

Context:
You are a helpful AI assistant.

Instruction:
Present your solution in the following format:
1. First, compute the sum of the two numbers, working from right to left using place values.
2. Then, for each place value, add the digits in the same place value column, and carry over if the sum is greater than 9.
3. Iterate this process from right to left until all place values are added.
4. Final Answer: Express using LaTeX notation `\boxed{answer}`.

Examples:
1. Compute `1996 + 126 = \boxed{?}`
`Let's solve 1996 + 126 step by step, working from right to left using place values.`

- For the ones place: 6 + 6 = 12. Write down 2 in the ones place and carry over 1 to the tens place.
- For the tens place: 9 + 2 + 1 = 12. Write down 2 in the tens place and carry over 1 to the hundreds place.
- For the hundreds place: 9 + 1 + 1 = 11. Write down 1 in the hundreds place and carry over 1 to the thousands place.
- For the thousands place: 1 + 1 = 2.
- Putting it all together: `2 * 1000 + 1 * 100 + 2 * 10 + 2 * 1 = 2000 + 100 + 20 + 2 = 2122.`

Therefore, `1996 + 126 = \boxed{2122}`.

2. Compute `1994 + 222 = \boxed{?}`
`Let's solve 1994 + 222 step by step, working from right to left using place values.`

- For the ones place: 2 + 4 = 6.
- For the tens place: 2 + 9 = 11. Write down 1 in the tens place and carry over 1 to the hundreds place.
- For the hundreds place: 2 + 9 + 1 = 12. Write down 2 in the hundreds place and carry over 1 to the thousands place.
- For the thousands place: 1 + 1 = 2.
- Putting it all together: `2 * 1000 + 2 * 100 + 1 * 10 + 6 * 1 = 2000 + 200 + 10 + 6 = 2216.`

Therefore, `1994 + 222 = \boxed{2216}`.

Question:
`%s + %s = \boxed{?}`

Figure 5: **Few-Shot Setting Prompt Template.** Example prompt template for few-shot addition tasks, providing context, instructions, examples, and question format for LLMs.

Prompt Template for Explain-and-Do Setting

Context:
You are a helpful AI assistant.

Instruction:
Present your solution in the following format:
1. First, comprehensively explain how to do addition with both positive integers.
2. Then, let's analyze the problem step by step following your explanation.
3. Final Answer: Express using LaTeX notation `\boxed{answer}`.

Question:
`%s + %s = \boxed{?}`

Figure 6: **Explain-and-Do Setting Prompt Template.** Example prompt template for explain-and-do addition tasks, providing context, instructions, and question format for LLMs.

Prompt Template for Symbolic Setting

Context:
You are a helpful AI assistant. Your task is to perform addition within a custom symbolic system in a simple and clear manner.

Symbolic System Definition:
This system comprises ten symbols: {u, d, a, i, h, v, e, y, r, c}. The addition operation (+) between these symbols is defined as follows:

```
u + u = u
d + u = d   d + d = a
a + u = a   a + d = i   a + a = h
i + u = i   i + d = h   i + a = v   i + i = e
h + u = h   h + d = v   h + a = e   h + i = y   h + h = r
v + u = v   v + d = e   v + a = y   v + i = r   v + h = c   v + v = du
e + u = e   e + d = y   e + a = r   e + i = c   e + h = du   e + v = dd   e + e = da
y + u = y   y + d = r   y + a = c   y + i = du   y + h = dd   y + v = da   y + e = di   y + y = dh
r + u = r   r + d = c   r + a = du   r + i = dd   r + h = da   r + v = di   r + e = dh   r + y = dv   r + r = de
c + u = c   c + d = du   c + a = dd   c + i = da   c + h = di   c + v = dh   c + e = dv   c + y = de   c + r = dy   c + c = dr
```

Instruction:
Present your solution in the following format:
1. **Align:** Arrange the two input strings vertically, aligning their rightmost symbols.
2. **Columnar Addition:** Starting from the rightmost column (least significant symbols), perform symbol addition using the provided definition.
3. **Carry-over:** If the result of a column's addition is a two-symbol sequence (e.g., ’da’), write down the second symbol (least significant) and carry over the first symbol to the next column on the left.
4. **Iteration:** Repeat steps 2 and 3, moving leftward column by column until all symbols have been added.
5. **Reasoning:** Keep your whole reasoning clear and simple.
6. **Output Format:** Write the final result in the `\boxed{?}` placeholder.

Examples:
1. Compute `dcce + dae = \boxed{?}`

Solution:
```
1. Columnar Addition (right to left):
- e + e = da (Write ’a’, Carry ’d’)
- c + a + d = da (Write ’a’, Carry ’d’)
- c + d + d = dd (Write ’d’, Carry ’d’)
- d + d = a
2. Result: adaa
3. Formatted Output: \boxed{adaa}
```

2. Compute `dcch + aaa = \boxed{?}`

Solution:
```
1. Columnar Addition (right to left):
- h + a = e
- c + a = dd (Write ’d’, Carry ’d’)
- c + a + d = da (Write ’a’, Carry ’d’)
- d + d = a
2. Result: aade
3. Formatted Output: \boxed{aade}
```

Your Task:
`Compute %s + %s = \boxed{?}`

Figure 7: **Symbolic Setting Prompt Template.** Example prompt template for symbolic addition tasks, providing context, symbolic system definition, instructions, examples, and question format for LLMs.

Task Type Models	Overall Acc. ZS	S	Δ	Position Add Acc. ZS	S	Δ	Carry-over Acc. ZS	S	Δ
Gemini2.0-pro-exp	94.88	14.21	-80.67	69.52	4.19	-65.33	77.36	7.07	-70.29
Gemini2.5-pro-exp (thinking)	99.16	55.99	-43.17	88.97	19.80	-69.17	88.49	24.56	-63.93
Gemini2.0-flash-exp	98.10	9.25	-88.85	73.83	1.21	-72.62	79.52	3.28	-76.24
Gemini2.0-flash-exp (thinking)	91.07	10.81	-80.26	86.09	2.89	-83.20	88.30	9.03	-79.27
Claude-3.5-Sonnet	99.81	7.51	-92.30	81.78	3.19	-78.59	90.28	6.92	-83.36
GPT-4o	93.39	9.59	-83.80	76.12	3.79	-72.33	79.55	6.73	-72.82
O1-preview	74.28	-	-	74.71	-	-	74.23	-	-
ERNIE-Speed-8K	73.78	0.29	-73.49	67.66	0.07	-67.59	70.89	0.21	-70.68
DeepSeek-V2.5	95.75	-	-	83.78	-	-	88.19	-	-
DeepSeek-V3	98.92	16.14	-82.78	78.55	11.98	-66.57	81.14	15.23	-65.91
DeepSeek-R1	97.39	-	-	70.99	-	-	80.58	-	-
DeepSeek-R1-Distill-Llama-70B	74.19	27.19	-47.00	68.91	42.94	-25.97	68.56	40.75	-27.81
DeepSeek-R1-Distill-Llama-8B	53.23	10.97	-42.26	45.54	39.55	-5.99	44.16	35.09	-9.07
DeepSeek-R1-Distill-Qwen-1.5B	58.16	0.66	-57.50	47.85	26.16	-21.69	47.16	20.79	-26.37
DeepSeek-R1-Distill-Qwen-7B	74.76	6.88	-67.88	65.38	33.41	-31.97	64.27	31.52	-32.75
Gemma2-2b-it	33.41	-	-	29.97	-	-	30.59	-	-
Gemma2-9b-it	66.34	1.45	-64.89	58.52	0.34	-58.18	60.44	0.44	-59.99
Gemma2-27b-it	83.65	2.62	-81.03	74.77	0.91	-73.85	76.68	0.91	-75.77
Llama2-7b-it	19.59	0.00	-19.59	20.44	0.01	-20.43	22.58	0.01	-22.57
Llama3-8B-it	32.95	0.24	-32.70	16.25	0.07	-16.18	15.89	0.20	-15.69
Llama3-70B-it	69.15	1.62	-67.53	59.84	0.39	-59.45	60.22	0.70	-59.52
Llama3.1-8B-it	43.34	0.57	-42.76	20.38	0.10	-20.27	21.96	0.25	-21.72
Llama3.1-70B-it	72.58	2.51	-70.07	60.13	0.52	-59.61	61.05	1.33	-59.71
Llama3.2-11B-it	35.13	0.53	-34.61	16.60	0.12	-16.48	17.35	0.26	-17.09
Llama3.3-70B-it	79.63	4.01	-75.61	73.82	0.77	-73.05	75.00	2.43	-72.57
Qwen1.5-7B-Chat	56.31	0.18	-56.14	46.78	0.05	-46.73	47.44	0.09	-47.34
Qwen1.5-72B-Chat	34.29	0.53	-33.75	62.28	0.09	-62.20	67.28	0.14	-67.13
Qwen2-7B-it	72.50	0.24	-72.26	60.03	0.05	-59.98	62.94	0.06	-62.88
Qwen2-72B-it	59.06	2.50	-56.56	82.82	0.21	-82.62	86.62	0.26	-86.36
Qwen2.5-1.5B-it	47.75	-	-	32.54	-	-	33.67	-	-
Qwen2.5-3B-it	70.27	-	-	54.49	-	-	57.98	-	-
Qwen2.5-7B-it	83.00	0.58	-82.41	71.39	0.11	-71.28	74.49	0.13	-74.37
Qwen2.5-14B-it	87.45	-	-	77.56	-	-	80.36	-	-
Qwen2.5-32B-it	95.15	-	-	90.41	-	-	91.28	-	-
Qwen2.5-72B-it	96.07	5.97	-90.10	88.19	2.09	-86.10	89.78	4.12	-85.67
QwQ-32B-Preview	70.59	11.12	-59.47	71.68	19.09	-52.59	73.22	20.71	-52.51
Eurus2-7B-SFT	83.21	0.42	-82.79	81.21	3.19	-78.02	82.28	6.87	-75.41
Eurus2-7B-PRIME	94.11	1.03	-93.08	91.59	3.10	-88.49	92.51	3.11	-89.40
qwen2.5-7b-dpo-sft-S	12.32	2.85	-9.47	9.31	0.58	-8.73	9.70	1.13	-8.57
qwen2.5-7b-dpo-sft-ZS	96.95	0.28	-96.67	84.48	0.29	-84.19	85.52	0.61	-84.91
qwen2.5-7b-dpo-S	50.73	24.10	-26.63	47.71	3.48	-44.23	48.40	6.37	-42.03
qwen2.5-7b-dpo-ZS	95.32	0.37	-94.95	86.23	1.17	-85.06	87.75	2.35	-85.40
qwen2.5-7b-sft-S	0.00	30.66	30.66	3.40	3.89	0.49	6.71	6.98	0.27
qwen2.5-7b-sft-ZS	97.17	0.00	-97.17	87.91	0.25	-87.66	89.51	1.26	-88.25

Table 7: **Complete Performance Analysis on Base and Extended Addition Tasks.** Per-model breakdown of performance (%) across standard numerical and symbolic representations, with evaluation of degradation (Δ) between formats. Results reveal systematic failures in abstracting arithmetic principles despite high numerical accuracy.