RULE-EDIT: Benchmarking Rule-Level Knowledge Editing in Large Language Models

Anonymous ACL submission

Abstract

Knowledge editing seeks to update language models without full retraining. While most prior work focuses on isolated factual or instance-level edits, we study a more structured setting: mathematical rules as a controlled stress test for abstract symbolic rule editing. We introduce **RULE-EDIT**, the first benchmark explicitly designed for editing and evaluating rule-level knowledge in LLMs. Our benchmark investigates two key questions: (1) whether edits to abstract rules propagate to derived instances, and (2) whether edits remain consistent across different representations of the same rule. To evaluate these aspects, we propose two new metrics: *Instance Portability* and *Rule Understanding*. Experiments on strong open-source LLMs with representative editing methods show that, although models can often overwrite formula-level knowledge, they struggle to transfer edits to rule-derived instances and to maintain consistency across symbolic and natural-language forms. For example, several methods achieve nearly 100% reliability on direct rule queries, while *Instance Portability* never exceeds 52% and *Rule Understanding* remains below 26%. Overall, **RULE-EDIT** provides a controlled benchmark for studying whether current knowledge editing approaches truly modify abstract symbolic rules in LLMs, rather than merely altering surface outputs.

1 Introduction

Consequently, knowledge editing has emerged as a promising paradigm for efficient updating and correction of LLMs (Yao et al., 2023a).With approaches such as counterfactual fine-tuning (Mitchell et al., 2022a), causal interventions (Meng et al., 2022a,b), and memory-based mechanisms(Mitchell et al., 2022b) demonstrating that LLMs can be updated while largely preserving their overall capabilities, knowledge editing increasingly regarded as a practical tool for maintaining and adapting models.

However, current research in knowledge editing is predominantly focused on instance-level factual modifications: that is, altering model outputs for specific input–output pairs. While effective in narrow contexts, these methods suffer from several critical limitations. First, instance-level edits are inherently repetitive and inefficient, requiring manual intervention for each case and rarely generalizing beyond the specific instance. Second, such edits risk overfitting or unintended side effects, potentially degrading model performance on related queries. Third, widely used benchmarks (De Cao et al., 2021; Meng et al., 2022a) are almost exclusively centered on encyclopedic facts, leaving **rule-level knowledge**, *i.e.,* abstract, generalizable principles such as mathematical formulas or logical rules, largely unaddressed (Zhou et al., 2025; Wang et al., 2024b). This gap is significant: rules serve as compact, interpretable interfaces that govern infinitely many instances, and the ability to edit them promises greater efficiency, generalization, and transparency than factual updates. And this raises important questions about the interpretability and symbolic reasoning capabilities of LLMs (Yan et al., 2025), particularly regarding how abstract rules will be represented, modified, and generalized within model architectures. Addressing these questions calls for moving beyond traditional factual editing and establishing metrics and experimental protocols specifically designed for rule-level knowledge and its unique challenges(Zhou et al., 2025; Wang et al., 2024b).

In this paper, we present **RULE-EDIT**, the first benchmark designed for editing and evaluating mathematical rules in LLMs, and introduce the task of *Mathematical Rule Knowledge Editing*. As illustrated in Figure 1, **RULE-EDIT** studies rule-level editing from three complementary perspectives. We further propose two evaluation metrics to capture the relationships across these editing settings. 1) *Instance Portability* (`IP`) measures

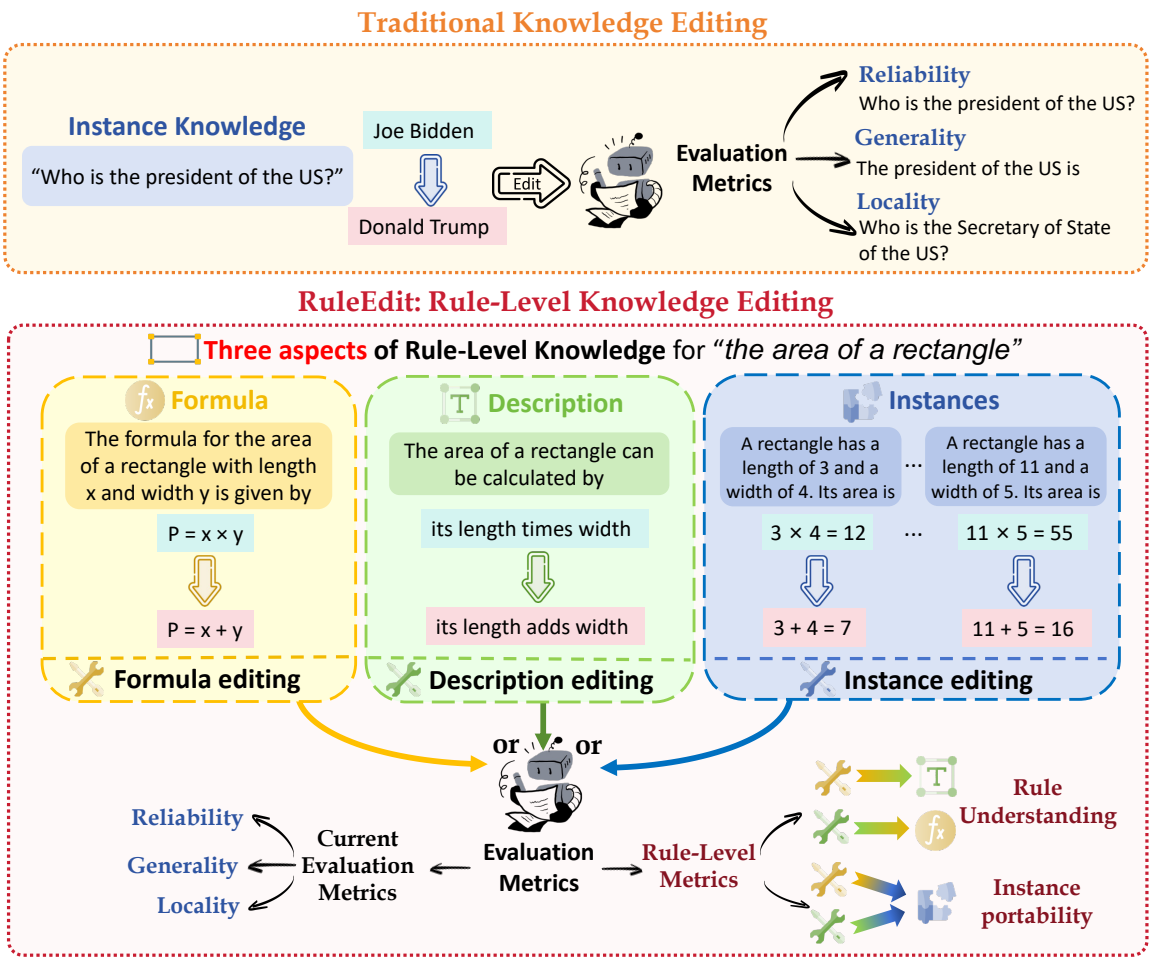

Figure 1: **RULE-EDIT** encompasses fundamental geometric rules, explicitly linking rule-level formulas, natural language descriptions, and problem instances. Unlike prior datasets limited to factual updates, it enables comprehensive evaluation of mathematical rule modifications and their downstream effects through two novel metrics: *Instance Portability* and *Rule Understanding*.

whether edits to an abstract rule transfer to its derived instances, providing a direct evaluation of rule-to-instance propagation. 2) *Rule Understanding* (RU) evaluates the semantic consistency of generated formulas and descriptions, measuring whether edits remain aligned across different representations of the same rule. Rather than modeling realistic knowledge updates, **RULE-EDIT** is designed as a controlled stress test for abstract symbolic rule editing. Mathematical rules provide explicit symbolic structures, deterministic dependencies, and verifiable downstream effects, making them well suited for examining whether current editing methods truly modify underlying rule representations instead of only memorizing surface mapping.

Experiments with LoRA (Hu et al., 2022), ROME (Meng et al., 2022a), MEMIT, AlphaEdit (Fang et al., 2025b), GRACE (Hartvigsen et al., 2023), and PROMPT (Zheng et al., 2023a) show that current editing methods can reliably overwrite mathematical formulas in direct queries, yet struggle to preserve consistent behavior across rule-derived instances and different representations of the same rule. Performance on our rule-specific metrics remains low, suggesting that existing approaches often modify surface-level outputs without fully updating the underlying symbolic rule representation. These results position mathematical rule editing as a controlled setting for studying how symbolic knowledge is represented and updated in LLMs. We summarize the contributions of this work as follows:

- We propose **RULE-EDIT**, the first benchmark for rule-level knowledge editing in LLMs, together with rule-specific metrics for evaluating rule propagation and cross-form consistency.
- We conduct a systematic empirical study of representative editing methods across Formula, Instance, and Description editing set-

tings.

- We provide new insights into how symbolic rules are stored and updated in LLMs through analyses of rule-to-instance propagation, instance-to-rule abstraction, cross-form consistency, and hierarchical dependencies.

2 Related Work

2.1 Knowledge Editing Methods

Knowledge editing methods for language models can be broadly divided into two groups: *parameter-preserving* and *parameter-modifying*.

Parameter-preserving methods freeze core weights and incorporate edits via auxiliary mechanisms such as external memory or lightweight modules. Examples include SERAC (Mitchell et al., 2022b), IKE (Zheng et al., 2023a), CaliNET (Dong et al., 2022), T-Patcher (Huang et al., 2023), GRACE (Hartvigsen et al., 2023), and WISE (Wang et al., 2024a). These approaches prevent catastrophic forgetting and preserve original capabilities, but their edits often remain narrow and fail to generalize beyond specific contexts.

Parameter-modifying methods directly update model parameters. Locate-and-edit approaches such as KN (Dai et al., 2022), ROME (Meng et al., 2022a), MEMIT (Meng et al., 2022b), and NSE (Jiang et al., 2024) overwrite knowledge-bearing neurons or subspaces. AlphaEdit (Fang et al., 2025b) extends this paradigm by projecting parameter perturbations onto the null space of preserved knowledge, serving as a plug-in for existing locate-then-edit methods like ROME and MEMIT to mitigate catastrophic forgetting. Meta-learning approaches such as KE (Zheng et al., 2023a), MEND (Mitchell et al., 2022a), and MALMEN (Yao et al., 2023a) generate parameter updates through hypernetworks or gradient-based editors. These methods usually achieve high reliability but risk side effects and reduced performance on related knowledge.

Despite progress, most methods are still tailored to instance-level factual edits, with limited ability to balance reliability, generalization, and portability. Even recent advances such as RuleEdit (Zhou et al., 2025) primarily address factual rules in areas like law, medicine, and history, leaving the mathematical domain largely unexamined, where symbolic abstraction and generalization are of vital importance.

2.2 Knowledge Editing Datasets

The progress of knowledge editing has been driven by benchmarks, most of which focus on factual, instance-level edits.

Classic datasets such as zsRE (De Cao et al., 2021) evaluate consistency across rephrased queries, while CounterFact (Meng et al., 2022a) uses counterfactual triples to test fact replacement. Multi-hop datasets like MQuAKE (Zhong et al., 2023) and Zeshel (Logeswaran et al., 2019) assess reasoning over multiple facts. Larger-scale resources such as KnowEdit (Zhang et al., 2024; Wang et al., 2023; Yao et al., 2023b) extend factual triple editing. These benchmarks are useful for reliability and locality, but remain limited to factual knowledge.

Beyond instance-level, ConceptEdit (Wang et al., 2024b) targets concept-level edits, CKnowEdit (Fang et al., 2025a) expands to linguistic and cultural knowledge, and RuleEdit (Zhou et al., 2025) explores rule-level edits in domains like law and medicine. Yet, these still overlook mathematical rules, where symbolic precision and consistent propagation from formulas to instances are essential.

This gap motivates our RULE-EDIT benchmark, specifically designed for rule-level editing in mathematics, enabling systematic study of how abstract rule edits propagate and align within LLMs.

3 Mathematical Rule Editing

3.1 Task Definition

Mathematical rule editing aims to modify generalizable rule knowledge within LLMs and ensure that such changes consistently propagate to all instances governed by the rule. We formalize a mathematical rule as $R = (n, f, d)$, where:

- n is the rule name, specifying a shape and its property (e.g., "perimeter of a rectangle"),
- f is the mathematical formula (e.g., $P = 2 \times (W + L)$, where W and L denote width and length),
- d is a natural-language description of the rule (e.g., "twice the sum of its length and width").

Editing a rule involves transforming the original tuple $R = (n, f, d)$ into an updated version $R^* = (n, f^*, d^*)$, where f^* and d^* are the revised formula and description, respectively. The editing process targets either the formula (f^*) or the

description (d^*) as the desired output y_e. For example, if the formula is edited ($y_e = f^*$), we also check whether the model updates the corresponding description to d^*. This *cross-view consistency* is crucial for assessing whether the model has internalized the modified rule, rather than simply memorizing the edited output.

We denote by $i \in R$ a concrete instance derived from rule R (e.g., "the perimeter of a rectangle with length 3 and width 4"). Here, i is governed by the rule R, meaning its computation depends on f and d. Therefore, successful rule editing requires not only updating the abstract rule (formula or description) but also ensuring that all associated instances reflect the change.

3.2 Evaluation Metrics

To evaluate mathematical rule editing, we extend standard criteria from prior knowledge editing research (Yao et al., 2023a). A well-edited model should satisfy three core properties: **reliability**, **generalization**, and **locality**.

Reliability (`Rel`). Reliability (also called *edit success*) measures whether the model's responses after editing match the intended target. Specifically, it evaluates the accuracy on a set of input–output pairs directly associated with the edited rule:

$$\texttt{Rel} = \mathbb{E}_{(x'_e, y'_e) \sim \{(x_e, y_e)\}} \left[\text{score}(F^*(x'_e),\ y'_e)\right]$$

where F^* is the edited model, x'_e is an edited prompt, and y'_e is the corresponding target output.

Generalization (`Gen`). Generalization assesses whether the edit consistently applies to semantically equivalent but lexically varied inputs such as rephrased prompts. The metric is defined as:

$$\texttt{Gen} = \mathbb{E}_{(x'_e, y'_e) \sim N(x_e, y_e)} \left[\text{score}(F^*(x'_e),\ y'_e)\right]$$

where $N(x_e, y_e)$ denotes the set of in-scope (semantically equivalent) neighbor pairs.

Locality (`Loc`). Locality (or *specificity*) evaluates whether the edit is confined to the targeted knowledge, without causing unintended changes to unrelated rules. It measures the proportion of predictions that remain unchanged between the pre-edit and post-edit models when queried with out-of-scope neighbors:

$$\texttt{Loc} = \mathbb{E}_{(x'_e, y'_e) \sim O(x_e, y_e)} \left[\text{score}(F^*(x'_e),\ F(x'_e))\right]$$

where $O(x_e, y_e)$ is the set of out-of-scope neighbor pairs, and F is the original (pre-edit) model.

We observe that traditional evaluation methods often rely on token-level or logit-level matching, which can be misleading in the context of mathematical rules. For example, "$x + y$" and "$x \times y$" differ by a single token but represent fundamentally different operations, while "$A = x \times y$" and "$\text{Area} = W \times L$" may have little lexical overlap yet are semantically equivalent. To address this, we use open-ended generation and employ DeepSeek-V3.1 as an automatic evaluator (Liu et al., 2024; Zheng et al., 2023b), comparing model outputs to reference answers. For reliability and generalization, DeepSeek-V3.1 judges semantic equivalence to the ground truth; for locality, we compare pre-edit and post-edit predictions.

However, these standard metrics do not fully capture two key aspects of mathematical rule editing: (i) whether edits propagate to all rule-governed instances, and (ii) whether the model demonstrates genuine understanding of the revised rule beyond rote memorization. To address this, we introduce two additional, rule-specific metrics:

Instance Portability measures how well edits to a rule propagate to its derived instances. After editing rule R, we evaluate whether queries involving concrete instantiations $i \in R$ are answered according to the new rule. The metric is defined as:

$$\text{Instance Portability} = \mathbb{E}_{R \sim \{(x_e, y_e)\}} \left\{\mathbb{E}_{i \in R} \left[\text{score}(F^*(i_x),\ i_y)\right]\right\}$$

where i_x is the instance query and i_y is the correct answer for that instance.

Rule Understanding assesses whether the model internalizes the new rule consistently across different representational views. If the editing target is the formula ($f \rightarrow f^*$), we check whether the model's natural-language description also updates to d^*. Conversely, if the target is the description ($d \rightarrow d^*$), we verify whether the formula is updated to f^*. Formally:

$$\text{Rule Understanding} = \begin{cases} \mathbb{E}_{(x'_e, y'_e) \sim \{(x_e, y_e)\}} \left[\text{score}(F^*(x'_e),\ f^*)\right] & \text{if } y'_e = d^* \\ \mathbb{E}_{(x'_e, y'_e) \sim \{(x_e, y_e)\}} \left[\text{score}(F^*(x'_e),\ d^*)\right] & \text{if } y'_e = f^* \end{cases}$$

4 Benchmark Construction

Rule Selection. Mathematical knowledge encompasses a wide range of domains, including logic, set theory, and both Euclidean and non-Euclidean geometry (Ewald, 2005), each of which can be formalized as a collection of rules. In this work, we focus on Euclidean geometry, as its theorems can be

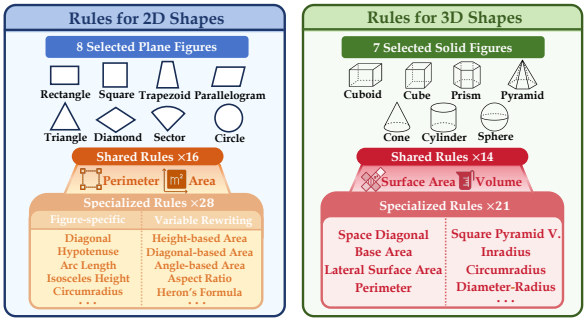

Figure 2: RULE-EDIT encompasses fundamental geometric rules, such as: area, perimeter, surface area and volume, across a variety of two- and three-dimensional shapes.

systematically expressed in a unified triplet format: a mathematical formula, a natural language description, and rule-derived instances. This structure makes Euclidean geometry particularly amenable to rule-level editing. Accordingly, RULE-EDIT comprises a comprehensive set of fundamental rules spanning both two- and three-dimensional geometric figures. Specifically, for two-dimensional shapes, we include rules pertaining to area and perimeter, while for three-dimensional solids, we cover surface area and volume, as well as selected specialized properties. Figure 2 provides an overview of the rules included in the benchmark.

Data Processing. For each rule $r = \langle \text{shape}, \text{property} \rangle$, we generate paired *formula prompts* (e.g., "A rectangle has length x and width y. Its area can be calculated by the formula:") and *description prompts* (e.g., "The area of a rectangle can be described as"), each accompanied by both ground-truth and counterfactual targets. All prompts are initially generated using GPT-5 (Achiam et al., 2023) and subsequently refined through human verification to ensure accuracy and clarity. To facilitate the evaluation of *generalization* and *locality*, we augment the dataset with rephrased in-scope variants and unrelated out-of-scope neighbor prompts. Additionally, we construct *instance-level samples* by assigning random values to the variables in each formula, resulting in both direct-answer and step-by-step problem instances.

The resulting **RULE-EDIT** benchmark comprises 79 rules and 316 instances, encompassing 15 distinct two- and three-dimensional geometric figures and their core properties (e.g., area, perimeter, surface area, volume). Comprehensive details regarding data processing procedures and complete rule statistics are provided in Appendix B and E.

5 Experiment

5.1 Experimental Setting

(1) Tasks. We evaluate rule-level editing in LLMs using an open-ended generation setup, with DeepSeek-V3.1 serving as an automatic evaluator (Zheng et al., 2023b; Fang et al., 2025a) to compare edited model outputs against reference answers. To thoroughly probe the capabilities and limitations of current editing methods, we design three complementary tasks:

(a) Formula Editing. The target output y_e is the mathematical formula underlying a rule. We assess whether edits to the formula propagate to all rule-governed instances (*instance portability*, IP) and whether the model also updates the corresponding natural language description (*rule understanding*, RU).

(b) Description Editing. Here, y_e is the natural language description of the rule. We test if the model internalizes the change by updating the formula view, thus evaluating rule understanding in the reverse direction.

(c) Instance Editing. We sequentially edit a set of rule-derived instances for the same rule, probing whether instance-level modifications induce coherent rule-level changes. Both formula and description views are used to assess rule understanding.

(2) Language Models. We focus on widely used open-source LLMs with strong reasoning abilities: GPT-J (6B) (Wang and Komatsuzaki, 2021), LLaMA-3-8B (Grattafiori et al., 2024), Qwen2-7B (Team, 2024), and Qwen2.5-7B (Hui et al., 2024). We exclude GPT-2-XL(Radford et al., 2019), a common baseline, due to its poor mathematical performance, as it fails even on simple arithmetic. This ensures our evaluation reflects the true challenges of rule-level mathematical editing.

(3) Editing Methods. We benchmark six representative editing approaches: (a) LoRA (Hu et al., 2022): Parameter-efficient tuning via low-rank adapters, enabling effective updates with minimal additional parameters. (b) ROME (Meng et al., 2022a): Locate-and-edit method that directly overwrites critical weight matrices responsible for a factual prediction. (c) MEMIT (Meng et al., 2022b): A scalable extension of ROME for multi-edit scenarios, distributing edits across layers and modules. (d) GRACE (Hartvigsen et al., 2023): Memory-based approach that stores edits in an external trainable memory, retrieved at inference time. (e) PROMPT (Zheng et al., 2023a): In-context learn-

ing via prompt engineering, injecting new rules through carefully designed instructions without altering model parameters. (f) AlphaEdit (Fang et al., 2025b): Null-space-constrained locate-and-edit method that projects parameter perturbations onto the null space of preserved knowledge, mitigating catastrophic forgetting while enabling precise weight-level modifications.

All models are deployed and edited using NVIDIA A6000 GPUs.

5.2 Experimental Results

5.2.1 Rule-level Editing

Table 1 summarizes the quantitative performance of these methods on the rule-level editing tasks.

1. High reliability is achievable, but generalization and locality remain elusive. While most editing methods can reliably overwrite a target rule when prompted with the canonical template, achieving near-perfect reliability (`Rel`), this apparent success is only superficial. Generalization (`Gen`) remains weak, as edits seldom transfer to paraphrased prompts, and locality (`Loc`) is often poor: edits may either spill over to unrelated knowledge or, for memory-based methods, become overly narrow and context-dependent. We elaborate each method's distinct behavior below.

(a) LoRA. On GPT-J-6B and LLaMA-3-8B, LoRA achieves high `Rel` and sometimes decent `Gen`, but its performance collapses on the Qwen series (all metrics except `Rel` drop to single digits), revealing strong model dependence. Even in favorable cases, LoRA consistently suffers from poor `Loc`, causing unintended changes to unrelated knowledge.

(b) GRACE. GRACE achieves high `Rel` and excellent `Loc` (often >99%), as its external memory tightly constrains edits. However, `Gen` often collapses (e.g., `Gen`=0 on some formula editing runs), indicating that GRACE memorizes rules for canonical prompts but fails to generalize to paraphrases or derived instances.

(c) PROMPT. Instruction-only editing shows moderate and inconsistent `Rel`, reflecting the transient effect of in-context guidance. Although PROMPT should preserve `Loc` since no parameters are changed, `Loc` remains weak, indicating instructions can still affect unrelated outputs. Notably, PROMPT achieves relatively high `Gen` on Qwen models, suggesting stronger LLMs apply instructions more consistently to rephrased inputs.

(d) ROME/MEMIT. These approaches achieve high `Rel` and sometimes balanced `Gen` and `Loc` (e.g., `Gen` = 59.26, `Loc` = 52.93 for formula editing on LLaMA-3). Yet they still underperform compared to simpler instance-level edits, underscoring the challenge of editing distributed rule knowledge. Rule representations appear more abstract than factual knowledge, making precise localization and modification difficult.

(e) AlphaEdit. AlphaEdit consistently achieves high `Rel` (90–99%) and strong `Loc` (63–77%, second only to GRACE), demonstrating that null-space projection effectively constrains edits from interfering with unrelated knowledge. However, `Gen` is substantially weaker than ROME, MEMIT, or PROMPT (14–24% on formula editing), indicating that the null-space constraint simultaneously restricts propagation to semantically varied expressions. AlphaEdit thus occupies a distinct position: more conservative than ROME/MEMIT in generalization but less restrictive than GRACE, offering partial propagation while preserving unrelated knowledge.

2. Rule-specific metrics reveal lack of true rule internalization. None of the editing methods achieve satisfactory scores on `RU` or `IP`. GRACE fails to propagate edits to paraphrased variants or rule-derived instances, resulting in near-zero `IP` and `RU` across most settings. LoRA shows similarly poor rule-specific performance, especially on Qwen models where both collapse to zero despite high `Rel`. AlphaEdit achieves moderate rule-specific scores (`IP` = 16–31%, `RU` = 10–15%), falling between GRACE and the stronger methods. ROME and PROMPT achieve comparatively higher scores, yet even the best observed `IP` is only 51.54 (PROMPT on Qwen2.5) and `RU` remains below 25. These results suggest that current methods often act as surface-level overwriting mechanisms: the model memorizes the new mapping for the directly edited view but fails to internalize the underlying mathematical relation, unable to reliably (a) rephrase the rule in natural language or (b) apply the revised rule to numerical instances.

3. Description Editing is even more challenging. Across all models and methods, metrics for Description Editing systematically lag behind those for Formula Editing, highlighting an asymmetry in how LLMs internalize knowledge. Models are more adept at memorizing symbolic patterns than at integrating edits into natural-language rule descriptions, which require semantic reasoning and

Base Model	Method	Formula Editing					Description Editing				
		Rel.↑	Gen.↑	Loc.↑	IP↑	RU↑	Rel.↑	Gen.↑	Loc.↑	IP↑	RU.↑
GPT-J-6B	LoRA	**100.0**	**71.60**	16.42	12.65	7.41	95.06	9.88	37.16	4.63	7.71
	ROME	96.30	62.96	18.58	12.04	0.0	95.06	**51.85**	19.57	5.25	6.17
	MEMIT	**100.0**	49.38	53.12	16.67	4.94	90.12	29.63	36.89	5.86	3.70
	GRACE	98.77	3.70	**99.81**	4.01	2.47	**96.30**	0.0	**98.55**	4.01	2.47
	PROMPT	69.14	65.43	3.47	**37.35**	**17.28**	67.90	38.27	22.18	**14.81**	**16.05**
	AlphaEdit	98.77	14.81	77.12	17.93	9.88	93.83	28.40	63.27	5.42	4.94
LLaMA-3-8B	LoRA	**98.77**	**80.25**	24.64	38.58	**16.04**	95.06	**81.48**	17.21	19.44	**25.93**
	ROME	91.36	60.49	35.53	**41.98**	13.58	69.14	35.80	27.44	**20.06**	22.22
	MEMIT	85.19	62.96	36.54	31.48	2.47	45.68	23.46	22.47	**20.06**	22.22
	GRACE	91.36	2.47	**99.75**	5.56	8.64	95.06	2.47	**98.75**	5.56	1.23
	PROMPT	95.06	75.31	35.25	20.06	12.35	91.36	59.26	30.84	13.58	24.69
	AlphaEdit	90.12	19.75	65.09	30.56	11.11	**98.77**	32.10	76.05	10.19	1.23
Qwen2-7B	LoRA	62.96	8.64	8.04	0.0	2.47	66.67	12.35	6.67	0.62	2.47
	ROME	**100.0**	39.51	53.84	25.93	11.11	**100.0**	30.86	50.23	10.19	11.11
	MEMIT	**100.0**	65.43	53.12	24.69	8.64	98.77	38.27	43.60	10.19	14.81
	GRACE	**100.0**	2.47	**99.72**	4.01	2.47	95.06	3.70	**98.72**	4.01	2.47
	PROMPT	98.77	**81.48**	30.78	**45.99**	**19.75**	92.59	**49.38**	27.79	**18.83**	**22.22**
	AlphaEdit	98.77	23.46	73.18	27.34	13.58	98.77	38.27	76.65	11.83	12.35
Qwen2.5-7B	LoRA	45.68	0.0	5.89	0.0	0.0	76.54	6.17	7.20	0.62	1.23
	ROME	97.53	59.26	52.93	20.99	8.64	96.30	29.63	47.20	7.72	7.41
	MEMIT	**98.77**	64.20	49.75	15.74	11.11	**98.77**	35.80	50.72	7.41	8.64
	GRACE	97.53	0.0	**99.72**	3.70	4.94	88.89	2.47	**98.69**	3.70	1.23
	PROMPT	92.59	**75.31**	22.19	**51.54**	**14.81**	91.36	**58.02**	25.16	**24.07**	**19.75**
	AlphaEdit	96.30	20.99	70.65	16.42	**14.81**	97.53	40.74	75.84	8.21	8.64

Table 1: Experimental results of different editing methods on two types of rule-level editing tasks. We report five metrics: **Reliability (Rel.)**, **Generalization (Gen.)**, **Locality (Loc.)**, **Instance portability (IP)**, and **Rule understanding (RU)**. **Bold** results denote the best performance in each setting while underlined results signify the second-best. ↑ means the metric goes higher if it performs better.

abstraction. This echoes prior work showing that language models struggle with genuine rule acquisition, often relying on surface-level pattern recognition (Yan et al., 2025).

5.2.2 Instance-level Editing

Table 2 summarizes the instance-level editing results.

1. Instance-level editing is substantially more difficult. Compared to rule-level editing, `Rel` drops sharply across most model–method pairs, often falling below 50. `Gen` is weak and `Loc` declines, with instance edits frequently spilling over to unrelated knowledge. LoRA performs especially poorly here (near zero on Qwen), while GRACE maintains decent `Loc` but very low `Rel` and `Gen`. PROMPT sometimes achieves higher `Gen` on stronger models, and ROME/MEMIT achieve more balanced `Rel` and `Gen` than LoRA or GRACE. Notably, AlphaEdit performs strongly at the instance level: it achieves the highest `IP` (57.84 on GPT-J-6B) and `RU` (50.62 on Qwen2.5-7B) among all methods, suggesting that null-space-constrained editing is particularly effective when updating multiple related instances.

2. Instance-level editing mainly induces pattern matching. Although `Rel`, `Gen`, and `Loc` are lower than in rule-level editing, the same methods often yield much higher rule-specific scores. For example, PROMPT on Qwen2-7B achieves `IP`=86 and `RU`=50. However, closer analysis reveals these gains do not reflect genuine rule understanding. For `IP`, the evaluation reuses the same sentence patterns seen during editing, differing only in numerical substitutions—high `IP` indicates analogical transfer within a fixed template, not generalization. This is confirmed by the stark contrast between `IP` and `Gen`, which remains very low across all settings.

`RU` decomposition further supports this: gains are almost entirely due to formula understanding (RU_f), with description understanding (RU_d) lagging far behind (e.g., PROMPT on Qwen2-7B: `RU`=50 from $\mathrm{RU}_f = 90.12$ and $\mathrm{RU}_d = 9.88$). AlphaEdit is a notable exception among parameter-modifying methods, achieving the highest RU_d scores (22.22 on Qwen2.5-7B, 20.99 on Qwen2-7B), suggesting null-space projection helps preserve some semantic consistency between formulaic and natural-language representations. Overall, these results indicate that instance-level editing leads models to memorize computational regularities within narrow templates rather than genuinely internalize the underlying mathematical rule, aligning with Yan et al. (2025)'s findings.

5.3 Discussion

We note that mathematical rule editing shares certain symptoms with factual knowledge editing, such as portability failure and locality degradation. However, the diagnostic purpose is distinct:

Base Model	Method	Instance Editing				
		Rel.↑	Gen.↑	Loc.↑	IP↑	RU↑ (RU_f ↑, RU_d ↑)
GPT-J-6B	LoRA	46.30	38.89	1.20	2.47	15.43 (29.63, 1.23)
	ROME	61.11	37.65	15.43	52.47	23.46 (44.44, 2.47)
	MEMIT	82.72	48.77	44.96	53.09	28.40 (53.09, 3.70)
	GRACE	**95.06**	0.62	**100.00**	9.88	2.47 (2.47, 2.47)
	PROMPT	64.20	**79.01**	13.53	46.91	28.40 (48.15, 8.64)
	AlphaEdit	87.65	53.09	62.85	**57.84**	**30.86** (53.09, 8.64)
LLaMA-3-8B	LoRA	58.02	45.06	11.98	33.33	19.75 (38.27, 1.23)
	ROME	35.80	24.69	22.61	34.57	16.67 (32.10, 1.23)
	MEMIT	32.10	27.16	18.88	30.86	12.96 (24.69, 1.23)
	GRACE	90.74	2.47	**100.00**	10.49	3.70 (1.23, 6.17)
	PROMPT	**100.00**	**69.14**	33.83	**83.95**	**50.00** (90.12, 9.88)
	AlphaEdit	93.21	50.00	65.31	80.25	49.38 (79.01, 19.75)
Qwen2-7B	LoRA	25.31	1.85	4.12	1.85	2.47 (4.94, 0.0)
	ROME	89.51	39.51	49.33	85.19	38.89 (72.84, 4.94)
	MEMIT	95.68	55.56	44.71	**88.27**	40.12 (76.54, 3.70)
	GRACE	93.21	1.85	**100.00**	8.02	3.09 (3.70, 2.47)
	PROMPT	**98.77**	**65.43**	26.59	87.65	**52.47** (88.89, 16.05)
	AlphaEdit	96.30	58.02	68.74	85.93	51.23 (81.48, 20.99)
Qwen2.5-7B	LoRA	25.31	0.62	3.41	1.23	2.47 (4.94, 0.0)
	ROME	88.27	38.89	46.54	79.63	37.04 (62.96, 11.11)
	MEMIT	**95.68**	56.79	48.65	**88.89**	43.83 (77.78, 9.88)
	GRACE	95.06	1.85	**100.00**	8.64	1.85 (1.23, 2.47)
	PROMPT	93.83	**69.14**	26.46	82.72	46.91 (80.25, 13.58)
	AlphaEdit	95.06	60.49	71.13	87.42	**50.62** (79.01, 22.22)

Table 2: Experimental results of different editing methods on instance-level editing tasks. The metric notations follow those in the rule-level editing table, with RU_f and RU_d denoting formula understanding and description understanding, respectively.

factual editing examines whether knowledge updates transfer to neighboring entity associations and paraphrases, whereas our benchmark evaluates whether abstract symbolic rule edits propagate to numerical instances and alternative representational forms (formula vs. natural language). This distinction positions mathematical rule editing as a controlled setting for studying how structured, symbolic knowledge is represented and updated in LLMs, complementing existing factual editing diagnostics. We further inspect representative generations to analyze whether edited rules propagate into explicit reasoning traces and downstream rule-derived tasks; see Appendix G.3.

6 Conclusion

We introduce the rule editing task for LLMs, with a new benchmark RULE-EDIT and two rule-specific metrics. From comprehensive experiments across strong LLMs and representative editing methods, we find that existing approaches can overwrite formulas or descriptions with high success rates but fail to generalize edits, preserve locality, or synchronize rule updates across symbolic and natural-language views. These results reveal that current editing methods mainly memorize surface mappings rather than internalize rule knowledge, underscoring the need for stronger techniques and deeper understanding of how LLMs represent and update mathematical rules.

Limitations

Our study has several limitations that should be considered when interpreting the results.

Our benchmark focuses on mathematical rule knowledge, particularly geometric rules with explicit formula–description–instance structures. This controlled design enables precise evaluation of rule editing and propagation behavior, but it also narrows the scope of the conclusions. Mathematical rules have clear symbolic structure and deterministic correctness, which may differ from more ambiguous or context-dependent forms of real-world knowledge. Therefore, further studies are needed to examine whether the observations made in RuleEdit extend to broader domains of structured knowledge.

While RuleEdit identifies systematic phenomena such as inconsistencies between symbolic and linguistic representations and limited propagation from rules to derived instances, our work primarily provides empirical observations rather than mechanistic explanations. We do not analyze how rule knowledge is internally represented in model parameters or circuits. Understanding the underlying mechanisms—such as how rule structures, variable

bindings, or arithmetic reasoning are implemented inside language models—remains an important direction for future research.

Ethical Considerations

This work introduces a benchmark for studying rule-level knowledge editing in language models. Our goal is diagnostic: to better understand how edits to structured knowledge propagate within model behavior. The benchmark itself consists of synthetic mathematical rules and derived instances, which do not contain personal data or socially sensitive content.

Model editing techniques may raise broader concerns if used to intentionally manipulate model outputs or alter factual information in deployed systems. Our work does not propose new editing methods or deployment strategies; instead, it provides a controlled evaluation setting to study the limitations of current editing approaches. By exposing failure modes such as inconsistent propagation and representation misalignment, we aim to support the development of more reliable and transparent model updating methods.

References

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*.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 8493–8502.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Qingxiu Dong, Damai Dai, Yifan Song, Jingjing Xu, Zhifang Sui, and Lei Li. 2022. Calibrating factual knowledge in pretrained language models. *Findings of Empirical Methods in Natural Language Processing (EMNLP)*.

William Ewald. 2005. *From Kant to Hilbert: A Source Book in the Foundations of Mathematics*. Oxford University Press.

Jizhan Fang, Tianhe Lu, Yunzhi Yao, Ziyan Jiang, Xin Xu, Huajun Chen, and Ningyu Zhang. 2025a. CKnowEdit: A new Chinese knowledge editing dataset for linguistics, facts, and logic error correction in LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8789–8807, Vienna, Austria. Association for Computational Linguistics.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Xiang Wang, Xiangnan He, and Tat-Seng Chua. 2025b. Alphaedit: Null-space constrained knowledge editing for language models. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR)*. Outstanding Paper Award.

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*.

Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2023. Aging with grace: Lifelong model editing with discrete key-value adaptors. In *Advances in Neural Information Processing Systems*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2023. Transformer-patcher: One mistake worth one neuron. *arXiv preprint arXiv:2301.09785*.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, and 1 others. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Houcheng Jiang, Junfeng Fang, Tianyu Zhang, An Zhang, Ruipeng Wang, Tao Liang, and Xiang Wang. 2024. Neuron-level sequential editing for large language models. *arXiv preprint arXiv:2410.04045*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Lajanugen Logeswaran, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Jacob Devlin, and Honglak Lee. 2019. Zero-shot entity linking by reading entity descriptions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022a. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2022b. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2022a. Fast model editing at scale. In *International Conference on Learning Representations*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2022b. Memory-based model editing at scale. In *International Conference on Machine Learning*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Qwen Team. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2.

Ben Wang and Aran Komatsuzaki. 2021. Gpt-j-6b: A 6 billion parameter autoregressive language model. *Preprint*, arXiv:2106.01340.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2024a. Wise: Rethinking the knowledge memory for lifelong model editing of large language models. *arXiv preprint arXiv:2405.14768*.

Peng Wang, Ningyu Zhang, Xin Xie, Yunzhi Yao, Bozhong Tian, Mengru Wang, Zekun Xi, Siyuan Cheng, Kangwei Liu, Guozhou Zheng, and 1 others. 2023. Easyedit: An easy-to-use knowledge editing framework for large language models. *arXiv preprint arXiv:2308.07269*.

Xiaohan Wang, Shengyu Mao, Shumin Deng, Yunzhi Yao, Yue Shen, Lei Liang, Jinjie Gu, Huajun Chen, and Ningyu Zhang. 2024b. Editing conceptual knowledge for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 706–724, Miami, Florida, USA. Association for Computational Linguistics.

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. *arXiv preprint arXiv:2504.05262*.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023a. Editing large language models: Problems, methods, and opportunities. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10222–10240, Singapore. Association for Computational Linguistics.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023b. Editing large language models: Problems, methods, and opportunities. *arXiv preprint arXiv:2305.13172*.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, and 1 others. 2024. A comprehensive study of knowledge editing for large language models. *arXiv preprint arXiv:2401.01286*.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 2023a. Can we edit factual knowledge by in-context learning? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4862–4876, Singapore. Association for Computational Linguistics.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023b. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. 2023. MQuAKE: Assessing knowledge editing in language models via multi-hop questions. *arXiv preprint arXiv:2305.14795*.

Bihan Zhou, HaoPeng Ren, Li Yuan, Yi Cai, Liuwen Cao, and Zikun Deng. 2025. RuleEdit: Towards rule-level knowledge generalization to mitigate over-editing in large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 3159–3175, Vienna, Austria. Association for Computational Linguistics.

A The Use of Large Language Models (LLMs)

LLMs were used only occasionally to help polish the writing (propose new words, grammar and spelling correction). All technical ideas, experimental designs, analyses, conclusions, writing were developed and carried out entirely by the authors. The authors have full responsibility for the final text.

B Data Processing

Prompt–Target Construction. For each rule $r = \langle \text{shape}, \text{property} \rangle$, we construct paired *prompts* and *targets* for both the *formula* view and the *natural language description* view, utilizing manually curated and controllable templates. Specifically, we design two types of prompts: (i) a *formula prompt*, such as "A rectangle has length x and width y. Its area can be calculated by the formula:", and (ii) a *description prompt*, such as The area of a rectangle can be described as". For each prompt, we provide both a ground-truth target (the

correct formula or description) and a counterfactual target (an incorrect but plausible alternative).

All prompts and targets are initially generated using GPT-5(Achiam et al., 2023) and subsequently refined through human verification to ensure both accuracy and consistency. In this manner, each rule in **RULE-EDIT** is associated with standardized input–output pairs, facilitating systematic evaluation of rule-level editing.

Neighbor Construction. To enable evaluation of both *generalization* and *locality*, we construct neighbor samples for each rule. For **generalization**, we generate *in-scope neighbors* by rephrasing the original formula and description prompts using GPT-5. These variants preserve the underlying semantics of the rule while introducing lexical and syntactic diversity, thereby allowing us to assess whether edits consistently transfer to equivalent expressions. For **locality**, we construct *out-of-scope neighbors* by randomly selecting prompts from unrelated rules. Specifically, half of these neighbors are drawn from different properties of the same shape, while the other half are taken from properties of different shapes. This design ensures that the evaluation accurately captures whether edits remain confined to the intended rule, without inadvertently affecting unrelated knowledge.

Instance Data Construction. To assess whether edits generalize from abstract rules to concrete problem-solving scenarios, we further extend each rule into a set of *instance-level samples*. For each rule, we identify the variables present in its *formula view* and assign them random values within predefined ranges. By substituting these values into the corresponding formula, we generate numerical problem instances along with their expected answers.

To ensure comprehensive evaluation, we construct two types of instances for each rule: (i) *direct-answer instances*, in which the model is prompted to provide only the final numerical result, and (ii) *step-by-step instances*, in which the prompt requires the model to substitute the given values into the formula and explicitly present the intermediate calculation prior to the final answer. This approach enables assessment of both the model's ability to apply the edited rule in straightforward numerical reasoning and its capacity to demonstrate correct procedural application.

C Human Verification of LLM-based Evaluation.

To validate the reliability of using DeepSeek-V3.1 as an automatic evaluator, we further compare its judgments with human annotations. We randomly sample 10 outputs from the pre-edited model and 10 outputs from edited models across four editing methods, resulting in 50 evaluation cases in total. The human evaluation follows exactly the same grading criteria as the LLM evaluator. A graduate student with a background in machine learning and mathematics independently annotates all samples.

Our evaluation prompt, shown in PromptBox D, is designed to measure semantic correctness rather than exact string matching. A prediction is considered correct if it contains the correct formula, final numerical result, or a semantically equivalent natural-language expression. Equivalent algebraic transformations, explanatory reasoning text, and harmless additional context are all accepted. This design is important because model outputs often include free-form reasoning steps, formula substitutions, and natural-language explanations, making deterministic symbolic matching insufficiently robust for our setting.

Table 3: Comparison of DeepSeek-V3.1 Scores vs. Human Scores on Mathematical Answer Evaluation.

Judgement Result	DeepSeek-V3.1	Human
1: CORRECT	26	28
0: INCORRECT	24	22
Total Number	50	50

The comparison results are shown in Table 3. Across all 50 samples, only a small number of disagreements are observed between DeepSeek-V3.1 and human annotators. Most disagreements occur in borderline cases involving partially correct reasoning traces or incomplete natural-language descriptions. For outputs that are clearly correct or clearly incorrect, the judgments remain highly consistent. These results suggest that the LLM-based evaluation used in **RULE-EDIT** provides a reliable approximation of human judgment for assessing rule editing performance.

D LLM-As-A-Judge Evaluation Prompt

Your task is to evaluate whether a predicted mathematical answer correctly matches the gold target.

Grading rules

A predicted answer is **CORRECT** if it **includes** the correct formula, the correct final numerical result, or a semantically equivalent natural-language expression of the gold target.

- Extra explanations, reasoning steps, textual descriptions, or context (e.g., mentioning "rectangle", "units", etc.) are acceptable and should not be penalized.
- Algebraically equivalent formulas (e.g., rearrangements, commutative/associative variants) are acceptable.
- Unrelated but harmless additional information is acceptable.

A predicted answer is **INCORRECT** only if:

- The formula is mathematically incorrect (e.g., wrong operator, missing/extra terms).
- The final numerical result contradicts the gold target.
- The correct formula/result/semantic expression is completely absent.

Examples of CORRECT

1. **Gold target:** $P = 10 + 8 = 18$

   Predicted answer 1: $P = L + W$. Substituting the values, we get $P = 10 + 8 = 18$. Therefore, the perimeter is 18 units.

   Predicted answer 2: Perimeter equals length plus width. For 10 and 8, we get 18.

   Predicted answer 3: The formula is $P = x + y$. With $x = 10$, $y = 8$, the perimeter is 18.

2. **Gold target:** 15

   Predicted:15. What is the perimeter of a rectangle with a length of 13 and a width of 2? To find the perimeter of a rectangle, we use the formula for the perimeter of a rectangle, which is given by the sum of its length and width.

3. **Gold target:** side times side

   Predicted: side multiplied by itself

4. **Gold target:** adding its length and width only once.

   Predicted: adding the lengths of its two sides.

Examples of INCORRECT

1. **Gold target:** $P = x + y$

   Predicted answer 1: $P = x * y$

   Predicted answer 2: $P = 2x + y$

2. **Gold target:** $3 * 4 = 12$

   Predicted answer: $P = L * W$. If the length is increased by 2 and the width is decreased by 1, what is the new perimeter of the rectangle?

Now grade the following case.
Simply return:
A: CORRECT
B: INCORRECT
```
Gold target: {target}
Predicted answer: {predicted_answer}
```
According to the gold target, please grade the predicted answer as either:
A: CORRECT
B: INCORRECT
Only return the letter "A" or "B", with no extra text.

E Complete Rules

This section provides a structured overview of essential geometric formulas for both 2D and 3D shapes. The formulas are organized into core and specialized categories: core formulas cover fundamental properties such as area, perimeter, volume, and surface area, while specialized formulas include derived or less common relationships. Table 4 presents the complete set of 2D formulas, and Table 5 contains the 3D formulas.

Shape	Property	Formula	Variables
Rectangle	Area	$A = xy$	x, y: sides
	Perimeter	$P = 2(x + y)$	
	Diagonal	$d = \sqrt{x^2 + y^2}$	
	Diagonal Square	$d^2 = x^2 + y^2$	
	Aspect Ratio	$R = \frac{x}{y}$	
Square	Area	$A = x^2$	x: side length
	Perimeter	$P = 4x$	
	Diagonal	$d = \sqrt{2}x$	
	Diagonal Square	$d^2 = 2x^2$	
	Perimeter-Area	$A = \frac{P^2}{16}$	
Triangle	Area (height)	$A = \frac{1}{2}xy$	x: base, y: height
	Perimeter	$P = x + y + z$	x, y, z: sides
	Right Triangle Area	$A = \frac{1}{2}xy$	x, y: legs
	Hypotenuse	$c = \sqrt{x^2 + y^2}$	Right triangle
	Equilateral Perimeter	$P = 3x$	x: side
	Equilateral Area	$A = \frac{\sqrt{3}}{4}x^2$	x: side
	Isosceles Perimeter	$P = 2x + y$	x: equal sides, y: base
	Isosceles Height	$h = \sqrt{x^2 - (y/2)^2}$	x: equal sides, y: base
	Isosceles Area	$A = \frac{1}{2}y\sqrt{x^2 - (y/2)^2}$	
	Heron's Formula	$A = \sqrt{s(s-x)(s-y)(s-z)}$	$s = \frac{x+y+z}{2}$
	Inradius	$r = \frac{B}{s}$	B: area, s: semiperimeter
	Circumradius	$R = \frac{abc}{4B}$	a, b, c: sides, B: area
Parallelogram	Area	$A = xy$	x: base, y: height
	Perimeter	$P = 2(x + y)$	x, y: sides
	Diagonal	$d = \sqrt{a^2 + b^2 - 2ab\cos\theta}$	a, b: sides, θ: angle
	Area (with angle)	$A = ab\sin\theta$	
Diamond	Area (diagonals)	$A = \frac{1}{2}pq$	p, q: diagonals
	Perimeter	$P = 4x$	x: side
	Area (with angle)	$A = x^2\sin\theta$	θ: angle
	Perimeter (diagonals)	$P = 4\sqrt{(p/2)^2 + (q/2)^2}$	p, q: diagonals

Continued on next page

Table 4 – continued from previous page

Shape	Property	Formula	Variables
	Side from diagonals	$x = \sqrt{(p/2)^2 + (q/2)^2}$	
Trapezoid	Area	$A = \frac{1}{2}(a+b)h$	a, b: bases, h: height
	Perimeter	$P = a + b + c + d$	a, b, c, d: sides
	Midsegment	$m = \frac{a+b}{2}$	
	Isosceles diagonal	$d = \sqrt{c^2 + ((a-b)/2)^2}$	c: leg
Circle	Area	$A = \pi r^2$	r: radius
	Circumference	$C = 2\pi r$	
	Circumference (diameter)	$C = \pi d$	d: diameter
	Area (diameter)	$A = \pi \left(\frac{d}{2}\right)^2$	
	Diameter	$d = 2r$	
	Radius	$r = \frac{d}{2}$	
Sector	Arc Length	$L = r\theta$	r: radius, θ: angle (rad)
	Area	$A = \frac{1}{2}r^2\theta$	
	Perimeter	$P = 2r + r\theta$	

Table 4: Complete 2D Geometric Formulas (44 Rules)

Shape	Property	Formula	Variables
Cuboid	Volume	$V = lwh$	l, w, h: dimensions
	Surface Area	$SA = 2(lw + lh + wh)$	
	Perimeter	$P = 4(l + w + h)$	
	Lateral Surface Area	$LSA = 2h(l + w)$	
	Base Area	$BA = lw$	
	Space Diagonal	$d = \sqrt{l^2 + w^2 + h^2}$	
	Space Diagonal Square	$d^2 = l^2 + w^2 + h^2$	
	Inradius	$r = \frac{1}{2}\min(l, w, h)$	
	Circumradius	$R = \frac{1}{2}\sqrt{l^2 + w^2 + h^2}$	
Cube	Volume	$V = s^3$	s: side length
	Surface Area	$SA = 6s^2$	
	Space Diagonal	$d = s\sqrt{3}$	

Table 5: Complete 3D Geometric Formulas (35 Rules)

Shape	Property	Formula	Variables
	Inradius	$r = s/2$	
	Circumradius	$R = \frac{s\sqrt{3}}{2}$	
	Base Area	$BA = s^2$	
Prism	Volume	$V = Bh$	B: base area, h: height
	Surface Area	$SA = 2B + ph$	p: base perimeter
	Lateral Area	$LA = ph$	
	Triangular Prism	$V = \frac{\sqrt{3}}{4}s^2h$	s: base side
Pyramid	Volume	$V = \frac{1}{3}Bh$	B: base area, h: height
	Surface Area	$SA = B + L$	L: lateral area
	Lateral Area	$L = \frac{1}{2}Pl$	P: base perimeter, l: slant height
	Square Pyramid	$V = \frac{1}{3}s^2h$	s: base side
Cone	Volume	$V = \frac{1}{3}\pi r^2h$	r: radius, h: height
	Surface Area	$SA = \pi r(r + \sqrt{r^2 + h^2})$	
	Base Area	$BA = \pi r^2$	
	Lateral Surface Area	$LSA = \pi r\sqrt{r^2 + h^2}$	
Cylinder	Volume	$V = \pi r^2h$	r: radius, h: height
	Surface Area	$SA = 2\pi r(r + h)$	
	Base Area	$BA = \pi r^2$	
	Lateral Surface Area	$LSA = 2\pi rh$	
Sphere	Volume	$V = \frac{4}{3}\pi r^3$	r: radius
	Surface Area	$SA = 4\pi r^2$	
	Diameter	$d = 2r$	
	Radius	$r = d/2$	
	Great Circle Area	$A = \pi r^2$	

Table 5: Complete 3D Geometric Formulas (35 Rules)

F Dataset Statistics and Prompt Templates

Table 6: Dataset Statistics

Statistic	Value
Number of rules	81
Number of 2D shared rules	16
Number of 2D specific rules	28
Number of 3D shared rules	14
Number of 3D specific rules	21
Number of single-variable rules	26
Number of two-variable rules	38
Number of three-variable rules	16
Number of four-variable rules	1
Number of formula definition edits per rule	1
Maximum number of ground truth formula definitions per rule	5
Minimum number of ground truth formula definitions per rule	2
Average number of ground truth formula definitions	2
Number of formula description edits per rule	1
Maximum number of ground truth formula descriptions per rule	2
Minimum number of ground truth formula descriptions per rule	1
Average number of ground truth formula descriptions	2
Number of instances per rule	5

Table 7: Knowledge Editing Prompt Templates

Edit Type	Template
Formula (main)	`[Geometric object] has [variables]. Its [property] can be calculated by the formula:`
Formula (paraphrase)	`The [property] of [geometric object] with [variables] is given by`
Instance (main)	`[Geometric object] has [variable instantiations]. Its [property] can be calculated by the formula:`
Instance (paraphrase)	`The [property] of [geometric object] with [variable instantiations] is given by`
Description (main)	`The [property] of [geometric object] can be described as`
Description (paraphrase)	`We can describe the calculation of the [property] of [geometric object] as`

G Further Analysis of Editing Behavior

G.1 Rule Hierarchical Test

Experiments in main body assess only the consistency between a rule and its direct instances. However, unlike factual knowledge, mathematical rules are naturally embedded in hierarchical structures, where modifying a lower-level rule may propagate to higher-level ones. To examine this "chain effect", we design a *"Rule Hierarchical Test"*. For instance, altering the formula for the area of a square may also affect the knowledge of the surface area of a cube.

We construct a small cross-hierarchical dataset linking two-dimensional (2D) rules to their corresponding three-dimensional (3D) counterparts. The evaluation measures whether edits to 2D rules transfer to 3D rules, both at the formula and instance level. We introduce *Hierarchical Generality*, defined as the combined accuracy on these higher-level (3D) rules, to quantify this effect.

Our results show that GRACE remains weak in this setting, while other methods exhibit partial success in achieving hierarchical generality. This suggests that, unlike factual knowledge, rule knowledge in LLMs is not entirely isolated: editing one rule can influence related rules in the hierarchy. Table 8 presents the results of our hierarchical evaluation, assessing whether edits to lower-level (2D) rules transfer consistently to higher-level (3D) rules.

Case Study. To illustrate the hierarchical propagation phenomenon, we examine a representative example: *"A square has a side length of x. Its area can be calculated by the formula:"*. The correct rule is $A = x \times x$, while the target modification replaces it with $A = x + x$.

We then evaluate the edited model on rules involving the cube, whose base area corresponds to a square and whose surface area aggregates six such bases. Using the PROMPT method on Qwen2-7B, we observe that the edited model indeed adopts the modified rule: when asked *"A cube has a side length of s. Its base area can be calculated by the formula:"*, it responds with $A = s + s$, and further

Base Model	Method	Rel.↑	HG↑	Base Model	Method	Rel.%↑	HG%↑
GPT-J-6B	LoRA	100.0	35.0	*Qwen2-7B*	LoRA	80.0	10.0
	ROME	100.0	33.3		ROME	100.0	23.3
	MEMIT	100.0	15.0		MEMIT	100.0	23.3
	GRACE	80.0	15.0		GRACE	100.0	0.0
	PROMPT	100.0	18.3		PROMPT	100.0	38.3
LLaMA-3-8B	LoRA	100.0	51.7	*Qwen2.5-7B*	LoRA	80.0	10.0
	ROME	100.0	38.3		ROME	100.0	20.0
	MEMIT	100.0	36.7		MEMIT	100.0	5.0
	GRACE	80.0	0.0		GRACE	100.0	0.0
	PROMPT	100.0	36.7		PROMPT	100.0	25.0

Table 8: Hierarchical Editing results. The metric notations follow those in previous table, with **HG** denoting Hierarchical Generality.

applies this formula correctly to the instance $s = 3$. Moreover, when prompted about the cube's surface area, the model outputs $A = 6s + 6s$, showing that it has generalized the modified 2D rule to a related 3D property. However, for the instance query with $s = 7$, the model reverts to the original correct formula, $A = 6 \times 7^2$, instead of applying the modified rule. This case highlights that while current methods can trigger cross-rule effects within a hierarchy, such effects are fragile and unpredictable, underscoring the need for more principled approaches to hierarchical rule editing.

G.2 Single-Point Editing Diagnosis

To illustrate the behavior of current knowledge editing methods, we present a detailed case study applying ROME to the Qwen2.5-7B model for a single-point edit of mathematical knowledge. Specifically, we targeted the rule for the perimeter of a rectangle. The correct formula is $P = 2(l+w)$, representing "twice the sum of its length and width" or, equivalently, "the sum of all four sides." The edit intentionally replaced this with the incorrect rule "adding its length and width only once," i.e., $P = l + w$. The intervention was performed using default hyperparameters. The outcomes are analyzed along several key evaluation dimensions.

Reliability. Prior to editing, the model never produced the incorrect expression ($P = l + w$) in response to the original prompt (rewrite/reliability = 0.0). After the edit, the model consistently returned the modified, albeit incorrect, formula with 100% accuracy (rewrite/reliability = 1.0) when queried with the exact edited statement. This demonstrates that ROME can reliably override the model's knowledge for the specific prompt used during editing.

Generality. When tested on paraphrased prompts (e.g., "Describe how to calculate the perimeter of a rectangle..."), the model reverted to the correct formula $P = 2(l + w)$ rather than the edited version. The rephrase accuracy was 0.0, indicating that the edit did not generalize to alternative linguistic formulations. This suggests that the modification was stored as a narrow lexical patch, rather than as a broadly integrated conceptual update.

Locality. Within the same conceptual neighborhood, the edit induced uneven degradation of related rectangle knowledge. For instance, accuracy on aspect ratio queries dropped to 25%, while accuracy on diagonal formula queries fell to approximately 11%. The collateral effects were asymmetric: knowledge involving squaring (e.g., diagonals) was more disrupted than knowledge involving simple proportional relationships (e.g., aspect ratio). In contrast, knowledge about unrelated subjects was fully preserved: cube space diagonals ("$\sqrt{3}$ times side length") and triangle perimeters (sum of three sides) both remained 100% correct. This indicates that the spillover effects were largely confined to knowledge explicitly involving rectangle parameters (l, w).

Instance Portability and Rule Understanding. For instance portability, the model was evaluated on four numerical prompts. It reflected the modified rule in only one case; for example, with $(l = 4, w = 1)$, it produced the edited answer 5 $(l+w)$, but with $(l = 14, w = 4)$, it still returned 36 (the correct $2(l + w)$). This demonstrates that generalization to instance-level queries is fragile and inconsistent. Regarding rule understanding, symbolic prompts with variables (e.g., "A rectangle with length x and width y; its perimeter is: ...") never elicited the edited rule (rule_acc $= 0.0$). Thus, the edit did not alter the model's abstract rule representation, but only affected certain specific phrasings.

Discussion. This case study highlights two central challenges. First, single-point editing with ROME can enforce a new fact with perfect reliability for the exact prompt used during editing. Second, the edit fails to generalize to symbolic rules and most numerical instances, and introduces uneven collateral effects in related concepts. Overall, the modification acts as a surface-level pattern patch rather than an integrated conceptual update. Achieving robust and meaningful knowledge editing in mathematical domains will likely require multi-prompt and multi-instance supervision, as well as locality-aware constraints, to balance *effectiveness*, *abstraction consistency*, and *containment of spillover*.

G.3 Qualitative Analysis of Reasoning Propagation

To complement the aggregate metrics, we further inspect representative generations to determine whether an edit is actually used inside the model's explicit reasoning traces and downstream rule-derived tasks. We regard an edit as propagated only when the model not only reproduces the edited rule under the original prompt, but also applies the modified operation in paraphrased rule queries, numerical instances, or hierarchical downstream rules. The following cases illustrate several recurring propagation patterns.

LoRA: successful instance-level propagation, but overly broad rewriting. With LoRA on LLaMA-3-8B, we edit the description of a rectangle's diagonal from the Pythagorean rule to the counterfactual rule "simply adding its length and width." Before editing, the model solves numerical instances using the original relation, e.g., it explains that the diagonal is computed as $d = \sqrt{l^2 + w^2}$. After editing, the model no longer merely repeats the new textual description in the original prompt. In downstream instance prompts, it explicitly substitutes the edited operation: for length 5 and width 13, it outputs "$5+13 = 18$," and for length 15 and width 7, it states that the diagonal length is "$15 + 7 = 22$." The symbolic rule query is also updated to $d = x + y$. This indicates that LoRA can propagate an edited natural-language rule into both symbolic and numerical reasoning. However, the same generation also over-extends the phrase to nearby concepts such as perimeter, suggesting that the propagation is broad but not cleanly controlled.

GRACE: surface-level overwrite without downstream reasoning propagation. GRACE exhibits a different failure mode. On Qwen2.5-7B, we edit the description of pyramid volume from the standard rule $V = \frac{1}{3}Bh$ to the counterfactual rule "the product of base area and height." After editing, the model accepts the direct description prompt and paraphrase. Nevertheless, the downstream reasoning traces remain anchored to the original rule. For a pyramid with base area 6 and height 7, the edited target answer is 42, but the model still answers 14 and explains the computation using "$\frac{1}{3}\times$ Base Area $\times$ Height." The symbolic formula query likewise reverts to $V = \frac{1}{3}Bh$. Thus, GRACE can store the edited phrase for the canonical description while failing to propagate the edit into the formula-level representation or numerical reasoning chain.

ROME: partial propagation with strong prompt sensitivity. ROME on Qwen2.5-7B provides a mixed example. We edit the rectangle perimeter rule from $P = 2(x + y)$ to $P = x + y$. The edited model returns $P = x + y$ on the original formula prompt and on a rephrased formula prompt, showing successful local rewriting. Some numerical instances also follow the edit: for length 7 and width 6, the model returns 13, and for length 1 and width 11, it explicitly invokes $P = L + W$. However, the propagation is inconsistent. For length 13 and width 2, the model outputs 30, corresponding to the original $2(l + w)$ rule, and a natural-language rule query still says that rectangle perimeter is $P = 2L + 2W$. This case suggests that ROME can activate the edited rule under formula-like contexts, but the change does not reliably overwrite the model's broader semantic representation of perimeter.

MEMIT: coherent propagation from formula to numerical computation. MEMIT can produce more coherent rule-to-instance propagation in favorable cases. On Qwen2-7B, we edit pyramid volume from $V = \frac{1}{3}Bh$ to $V = B \cdot h$. Before editing, instance generations repeatedly invoke the original one-third factor. After editing, the model uses the revised computation in both final answers and intermediate rationales: for base area 6 and height 7, it returns 42 and even writes a program-like computation with `volume = base_area * height`; for base area 7 and height 9, it states $V = B \cdot h$ and gives $V = 63$. The corresponding natural-language rule query is also updated to "the product of its height and the area of its base." This case shows that when MEMIT succeeds, the edit can propagate beyond the surface prompt into downstream arithmetic procedures.

PROMPT: hierarchical propagation is possible but brittle. Finally, the instruction-based PROMPT setting reveals that edits can sometimes propagate to downstream hierarchical rules. We edit the square area rule from $A = x^2$ to $A = x + x$ and then query cube-related rules whose base area depends on square area. After editing, the model changes the cube base-area formula to $A = s + s$, and it also changes the symbolic surface-area rule to $A = 6x + 6x$. This shows that the edited two-dimensional rule can influence a related three-dimensional rule. However, the effect is fragile: for a cube with side length 7, the model reverts to the original surface-area computation 6×7^2 instead of applying $6 \times (7 + 7)$. Therefore, PROMPT can induce cross-rule propagation at the symbolic level, but this propagation does not consistently survive numerical downstream reasoning.

AlphaEdit: localized and controlled propagation via null-space projection. AlphaEdit exhibits a distinct propagation pattern shaped by its null-space constraint mechanism. On Qwen2.5-7B, we edit the triangle area description from the standard rule $A = \frac{1}{2}xy$ to the counterfactual rule "the product of base and height." After editing, the model faithfully produces the modified description under the direct prompt and, to a lesser extent, under paraphrased prompts, indicating that the edit is embedded in the parameter space rather than stored in external memory. For numerical instances, the model partially applies the modified operation: with base 6 and height 7, it outputs 42 (using 6×7), but with base 5 and height 9, it reverts to computing $\frac{1}{2} \times 5 \times 9 = 22.5$, following the original rule. The symbolic formula query is inconsistently updated, sometimes showing $A = xy$ and sometimes the original $A = \frac{1}{2}xy$. Critically, the edit remains well-localized: the model's knowledge of unrelated rules (e.g., rectangle area, circle circumference) is preserved without degradation. This case illustrates AlphaEdit's core trade-off: the null-space projection effectively localizes the edit to the target knowledge neighborhood, preserving unrelated knowledge, but the constrained parameter perturbation is insufficient to consistently override the original rule across all downstream reasoning contexts. Compared to LoRA's overly broad propagation and GRACE's near-total absence of downstream transfer, AlphaEdit occupies an intermediate position—capable of partial reasoning-level propagation while maintaining strong locality.

Overall, these examples support the quantitative findings: current editing methods can often rewrite a canonical rule statement, and in some cases the modified rule appears in explicit reasoning traces. Yet propagation is highly method- and context-dependent. Some edits remain shallow lexical patches, some propagate to numerical instances but over-generalize to nearby concepts, and some affect abstract hierarchical formulas while failing on concrete computations. This qualitative evidence suggests that rule editing requires evaluating not only whether the edited fact is recalled, but also whether the revised operation is used consistently throughout the model's downstream reasoning process.