SHAPE of Chain-of-Thought in Math Reasoning

Jonghyun Song [1] **Sangjun Song** [1] **Minjae Oh** [1] **Haesung Pyun** [1] **Sungsik Lee** [2] **Yohan Jo** [1]

Abstract

Large language models (LLMs) achieve strong performance on mathematical reasoning benchmarks, yet the mathematically meaningful skills underlying their reasoning remain underexplored. We introduce SHAPE, a framework that analyzes Chain-of-Thought (CoT) trajectories through two lenses developed in mathematics education: (1) semantic spaces: the model's evolving mathematical interpretations of a problem (e.g., algebraic, geometric), and (2) heuristics: the specific mathematical actions taken within those spaces (e.g., simplifying the problem, working backward). We first use SHAPE to analyze the reasoning patterns of various models. Our findings reveal that the mathematical heuristics employed by a model better explain final answer correctness than traditional CoT features. Furthermore, models are likely to reach correct solutions by concentrating their reasoning effort within a few semantic spaces rather than exploring many disparate ones—a pattern consistent with human behavior. Next, we utilize the SHAPE lens to evaluate whether post-training truly enhances mathematical proficiency. We find that reinforcement learning induces mode-seeking in heuristic usage. Lastly, we post-train LLMs by promoting diverse heuristics and demonstrate its effectiveness in improving accuracy. Overall, SHAPE provides a theoretically-grounded diagnostic framework for decoding LLM reasoning and offers a new path toward post-training LLMs for math reasoning. The code for our model is available at https://github.com/holi-lab/SHAPE-of-CoT

[1] Graduate School of Data Science, Seoul National University, Seoul, Korea [2]Department of Mathematics Education, Seoul National University, Seoul, Korea. Correspondence to: Yohan Jo <yohan.jo@snu.ac.kr>.

The 3rd *AI for Math Workshop* at the 43rd International Conference on Machine Learning (ICML), Seoul, South Korea, 2026. Copyright 2026 by the author(s).

1. Introduction

Recent large language models (LLMs) achieve strong performance on mathematical reasoning benchmarks, often by generating long Chain-of-Thought (CoT) trajectories (Hendrycks et al., 2021; Wei et al., 2022; Jaech et al., 2024; Yang et al., 2025a). However, final-answer accuracy offers limited insight into *how* a model organizes its solution. Existing CoT analyses capture useful surface properties—length (Wu et al., 2025b; Su et al., 2025), self-revision markers such as "wait" or "aha" (Guo et al., 2025), broad reasoning episodes such as planning and verification (Gandhi et al., 2025; Marjanovic et al., 2026; Li et al., 2025b;a), or structural patterns such as graphs and trees (Jiang et al., 2025; Xiong et al., 2025; Zhang et al., 2025)—but none track the specific mathematical interpretation under which a model is operating. As illustrated in Figure 1, a model may interpret a problem algebraically, abandon that interpretation to explore numerical cases, and then return to the algebraic one; all within a single correct solution. Understanding which strategies a model employs and how it moves between interpretations offers a principled basis for diagnosing reasoning failures and guiding improvements, yet remains largely unexplored.

Mathematics education research has long grappled with this question: how solvers organize mathematical problems (Pólya, 1945; Schoenfeld, 1985). The field has converged on two core axes—the solver's current mathematical interpretation of the problem, termed a *semantic space* (Newell et al., 1972; Favier & Dorier, 2024), and the purposeful actions taken within that interpretation, termed *heuristics* (Favier & Dorier, 2024; Koichu et al., 2007; Rott, 2014). LLM-generated CoT trajectories can be analyzed through these two axes to infer how a model frames, pursues, and reorganizes a problem. To that end, we introduce SHAPE (Semantic-space and Heuristic Analysis for Problem-solving Evolution), a framework that represents LLM reasoning as a sequence of semantic spaces and heuristics. SHAPE measures how heuristic activity is distributed across semantic spaces and how frequently the model transitions between them, capturing whether a model reasons coherently or diffuses its effort across interpretations without committing to any. We operationalize SHAPE through an automated pipeline that scales heuristic tagging and semantic-space tracking to thousands of trajec-

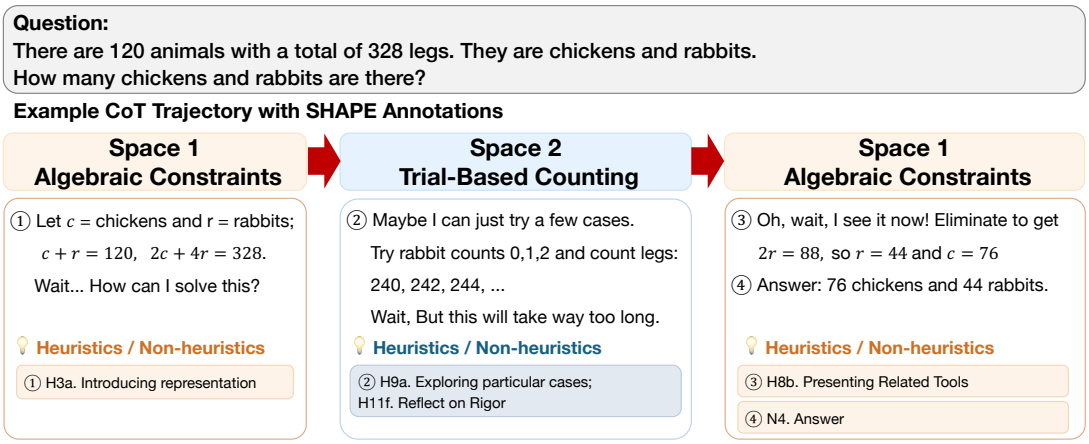

Figure 1. Illustrative example of SHAPE annotation applied to a synthetic CoT trace. The solver first sets up an algebraic formulation (Space 1), introducing a system of equations. It then abandons this approach and switches to a trial-based counting strategy (Space 2). Finally, it returns to the algebraic formulation (Space 1) and solves the system. We refer to each problem-solving interpretation as a *semantic space*—a distinct mathematical interpretation defined by the objects, goals, and constraints the model adopts. Within each space, individual steps are tagged as *heuristics*—purposeful mathematical actions such as introducing a representation (H3a) or exploring particular cases (H9a)—or as non-heuristic steps such as stating the answer (N4). The arrows mark transitions between these spaces. For the full heuristic taxonomy, see Appendix A.

tories across models.

We demonstrate the utility of SHAPE by (1) analyzing the reasoning patterns of various LLMs, (2) examining the effect of post-training on reasoning patterns, and (3) directly enhancing post-training. First, we validate SHAPE by showing that heuristic-level features of reasoning predict answer correctness more reliably and consistently across models than existing CoT features such as length, lexical markers, and episode labels. Furthermore, we find that successful reasoning is associated with coherent, focused engagement within a small number of semantic spaces, whereas incorrect trajectories tend to scatter heuristic activity across many disparate spaces. Second, applying this lens to post-training, we find that when problems require a fundamentally different solution approach than those the model already handles well, RL-trained models fail to reorganize their semantic-space engagement and instead oscillate between already-visited spaces. In addition, post-training induces mode-seeking in heuristics usage, concentrating successful trajectories into a narrower heuristics distribution. Third, motivated by these findings, we directly incorporate mathematical heuristics into planning during reinforcement learning with verifiable rewards (RLVR). We demonstrate this method improves task performance.

2. SHAPE: Framework and Pipeline

2.1. SHAPE Framework

Theoretical Background *Semantic-space and Heuristic Analysis for Problem-solving Evolution* (SHAPE) instantiates an analytical approach developed in mathematical problem-solving research, where reasoning is studied not only through final answers but through observable records produced during solving. In human studies, such records include verbalizations elicited through think-aloud protocols (Ericsson & Simon, 1993; Koichu et al., 2007; Carlson & Bloom, 2005) and audiovisual records of problem-solving actions (Favier & Dorier, 2024). For LLM mathematical reasoning, the model-generated CoT trajectory can be thought of as the corresponding observable record: a textual trajectory of stated intermediate moves, including reformulations, transformations, verifications, and conclusions. SHAPE uses this trajectory to analyze what mathematical actions the model takes, what underlying interpretation of the problem those actions suggest, and how this interpretation evolves over the course of solving. SHAPE is built on two complementary concepts: *heuristics* (Koichu et al., 2007; Favier & Dorier, 2024; Rott, 2014) and *semantic spaces* (Favier & Dorier, 2024; Newell et al., 1972). Heuristics refer to the visible mathematical actions expressed in the CoT trajectory, while semantic spaces describe the mathematical interpretation of the problem solver inferred from those actions.

Heuristics Heuristics are problem-solving devices—such as strategies, tactics, or local operations—that guide the solving process (Rott, 2014). Examples include simplifying the problem, working backward, introducing auxiliary objects, changing representation, making cases, or arguing by contradiction (Pólya, 1945; Schoenfeld, 1985; Koichu et al., 2007; Favier & Dorier, 2024). We consolidate prior taxonomies proposed for different mathematical problem-solving contexts (Pólya, 1945; Koichu et al., 2007;

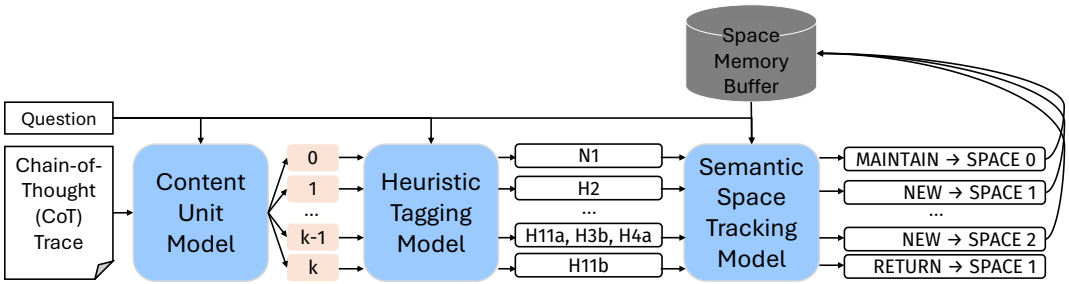

Figure 2. Overview of the automated SHAPE analysis pipeline. Given a Chain-of-Thought (CoT) trajectory, the pipeline first segments the text into content units corresponding to heuristic-bearing spans. It then assigns one or more heuristic labels to each unit using a tagging model. Finally, a semantic-space state tracking model classify the model's mathematical interpretation change into MAINTAIN, NEW, or RETURN.

Favier, 2022; Posamentier & Krulik, 2008) into a unified set designed for automated annotation of LLM-generated CoT trajectories. The full taxonomy is provided in Appendix A.

Semantic Spaces A semantic space is the solver's mathematical interpretation of the problem. It captures the solver's current interpretation of the problem: which constraints and goals are considered, and which actions are seen as available for progress (Newell et al., 1972; Favier & Dorier, 2024). For example, in Figure 1, the solver first frames the problem algebraically by setting up a system of equations (*Algebraic Constraints*), then shifts to enumerating solutions (*Trial-Based Counting*), before returning to the algebraic frame to complete the solution. In SHAPE, semantic-space labels are inferred by reading sequences of heuristic actions in the CoT trajectory, as semantic spaces are usually not explicitly stated.

2.2. Automated SHAPE Analysis Pipeline

Manual SHAPE annotation provides an adjudicated reference set, but it does not scale to the large number of trajectories we analyze. We therefore build an automated pipeline that converts raw CoT trajectories into structured SHAPE trajectories.

Gold Standard and Annotator Model Selection To ground SHAPE in actual LLM reasoning behavior, we construct a gold set of annotated CoT trajectories. We curate 48 CoT trajectories on the MATH-Perturb dataset (Huang et al., 2025) from four models spanning thinking and non-thinking modes as well as different model sizes: Qwen3-30B-A3B-Instruct, Qwen3-30B-A3B-Thinking, Qwen3-8B (Yang et al., 2025a), and Nemotron-Cascade-8B (Wang et al., 2025). Each trajectory is segmented into *content units*—the smallest spans that can be coherently annotated with heuristics (Koichu et al., 2007)—yielding 1,598 units from 8,334 sentences in total. Because heuristic tagging is multi-label and inherently interpretive, we adopt a consensus protocol. Four authors, including a graduate researcher in mathematics education, annotated the trajectories and resolved each unit through discussion until reaching a shared interpretation. This procedure follows mathematics education research in treating problem-solving analysis as interpretive annotation, where discussion is used to ensure objectivity in coding decisions (Koichu et al., 2007). Detailed tagging procedure is provided in Appendix B.

We then evaluate candidate annotator models against this gold set using weighted F1 for unit-level agreement and macro F1 for class-level agreement. Grok-4.1-Fast (xAI, 2025) and Qwen3.5-27B (Qwen, 2026) achieve the strongest closed-source and open-source performance, respectively, and serve as our primary annotator models across all pipeline stages. Unless otherwise noted, all subsequent SHAPE annotations are performed using Qwen3.5-27B.[1] Full performance, Cohens kappa scores, and semantic-space tracking validation are reported in Appendix C.

Pipeline Stages Our automated pipeline converts raw CoT trajectories into structured SHAPE trajectories through three stages: content-unit segmentation, heuristic tagging, and semantic-space tracking (Figure 2). The first two stages follow standard prompting-based annotation. As semantic-space tracking requires a more structured approach, we prompt the tracking model to assign NEW, RETURN, or MAINTAIN to content units that include a representation-changing heuristic[2]; NEW opens a fresh semantic space, MAINTAIN continues the current one, and

[1] Annotating 445 trajectories with Grok-4.1-Fast costs approximately $100, which is impractical for large-scale annotation.

[2] H1. Changing the register of semiotic representation; H2. Cognitive reinterpretation; H3. Introduce symbolic representation, formalization, and structural augmentation; H5. Wishful thinking (simplify / reduce the problem and conditions); H8. Analogy and presenting related theorems; H11. Verification and looking back. These correspond to space-changing heuristics identified in (Favier & Dorier, 2024).

when RETURN is selected, the semantic space tracking model $\mathcal{A}$ jointly determines the target space ID from the memory buffer $\mathcal{M}$ in a single call. The other units are considered to be remaining in the current semantic space. The exact algorithm (Algorithm 1) and the prompts for all three stages are provided in Appendix D.

2.3. Metrics

Based on the semantic spaces and heuristics annotated within our framework, we characterize the distributional patterns of heuristic usage across semantic spaces in the reasoning trajectories of LLMs. During the reasoning process, a model may concentrate its heuristic usage within a single semantic space, engaging in deep exploration, or distribute its heuristics across multiple semantic spaces, resulting in a more shallow exploration. In other words, different reasoning trajectories can be understood in terms of how strategic effort (i.e., heuristics) is allocated across semantic spaces. This distinction is grounded in mathematics education, where analogous depth-versus-breadth profiles have been used to characterize students' problem solving (Favier & Dorier, 2024).

To formalize this distinction, we first represent a CoT trajectory of T content units via two annotation layers: a semantic-space sequence $\mathbf{S} = (s_1, s_2, \ldots, s_T)$, where each $s_t \in \mathcal{I}$ is the semantic space assigned to content unit t, and a heuristic sequence $\mathbf{H} = (H_1, H_2, \ldots, H_T)$, where $H_t \subseteq \mathcal{H}$ is the set of heuristics assigned to the same unit. Here, $\mathcal{I} = \{1, \ldots, M\}$ is the set of distinct semantic space IDs observed in the trajectory, and $\mathcal{H}$ denotes the full heuristic taxonomy, which is provided in Appendix A. From $\mathbf{S}$ and $\mathbf{H}$, we derive two complementary distributions that capture different aspects of heuristic allocation across the trajectory. The first, which we call the *space distribution*, aggregates heuristic counts by semantic space ID: $q(i) = \sum_{t=1}^{T} \mathbf{1}[s_t = i]\,|H_t| \,/\, \sum_{t=1}^{T} |H_t|$ for $i \in \mathcal{I}$. The second, the *segment distribution*, instead operates at the level of contiguous segments: $p(k) = \sum_{t \in R_k} |H_t| \,/\, \sum_{t=1}^{T} |H_t|$ for $k \in \{1, \ldots, K\}$, where R_k is the k-th maximal contiguous segment of a single semantic space ID in $\mathbf{S}$. The distinction matters because a model may revisit the same semantic space multiple times; the space distribution merges these visits, whereas the segment distribution treats each contiguous visit separately. We then quantify each distribution using its corresponding *effective number* (Jost, 2006), defined as the exponential of its entropy. Intuitively, this measures the number of semantic spaces (or segments) that are meaningfully utilized, accounting for how heuristic activity is distributed across them. Formally,

$$N^{\text{eff}}_{\text{space}} = \exp\left(-\sum_{i \in \mathcal{I}} q(i) \log q(i)\right) \tag{1}$$

$$N^{\text{eff}}_{\text{trans}} = \exp\left(-\sum_{k=1}^{K} p(k) \log p(k)\right) - 1 \tag{2}$$

For instance, in Figure 1, $\mathbf{S} = (1, 2, 1)$ with heuristic counts $(1, 2, 1)$ gives $q = (1/2,\ 1/2)$ and $N^{\text{eff}}_{\text{space}} = 2$; the segment distribution $p = (1/4,\ 1/2,\ 1/4)$ yields $N^{\text{eff}}_{\text{trans}} \approx 1.83$. $N^{\text{eff}}_{\text{space}}$ and $N^{\text{eff}}_{\text{trans}}$ capture complementary reasoning structure. A high $N^{\text{eff}}_{\text{space}}$ indicates that heuristic activity is spread across many semantic spaces, whereas a high $N^{\text{eff}}_{\text{trans}}$ indicates frequent transitions between segments. For example, a trajectory with low $N^{\text{eff}}_{\text{space}}$ but high $N^{\text{eff}}_{\text{trans}}$ suggests that the model repeatedly alternates among a narrow set of semantic spaces without broadening its exploration.

The preceding metrics describe how heuristic activity is spread across semantic spaces, but they do not reveal which heuristics are used or how often. To capture this, we introduce the *heuristic frequency distribution* $u(h) = \sum_{t=1}^{T} \mathbf{1}[h \in H_t] \,/\, \sum_{t=1}^{T} |H_t|$ for $h \in \mathcal{H}$ captures the overall usage share of each heuristic across the trajectory.

3. Analysis on the Reasoning Patterns of LLMs

In this section, we first validate SHAPE by demonstrating that heuristic features predict answer correctness better than conventional CoT representations (e.g., CoT length, self-revision markers), and then analyze the structural reasoning patterns of various LLMs revealed by semantic-space metrics.

Table 1. Correctness prediction under 5-fold stratified cross-validation. "Length + reasoning" includes CoT length together with reasoning features (reasoning-token count and proportion), while "Self-revision" includes self-revision features derived from lexical self-revision markers. Within SHAPE, H denotes the 11 heuristic categories (H1–H11) and N the non-heuristic category.

Feature Set	AUROC ↑	# Features
Length	0.504 ±0.03	1
Length + reasoning	0.503 ±0.03	3
Self-revision	0.618 ±0.03	3
ThinkARM	0.618 ±0.02	8
SHAPE (H)	0.653 ±0.02	11
SHAPE (H+N)	**0.664 ±0.02**	12

3.1. Heuristic-Level Features Outperform Existing CoT Representations

To test whether heuristic features provide richer signals than existing CoT analysis methods, we train logistic regression models to predict whether a CoT trajectory leads to a correct answer, using each baseline feature set as input.

Table 2. Descriptive statistics of SHAPE metrics by correctness. $N^{\text{eff}}_{\text{space}}$: effective number of semantic spaces; $N^{\text{eff}}_{\text{trans}}$: effective number of semantic-space transitions; transition ratio ρ: measuring the intensity of revisiting semantic space. (C), (I), and (O) denote correct, incorrect, and overall trajectories.

	$N^{\text{eff}}_{\text{space}}$			$N^{\text{eff}}_{\text{trans}}$			ρ			
Model	(C)	(I)	(O)	(C)	(I)	(O)	(C)	(I)	(O)	Acc.
Open-source reasoning models with full traces										
Qwen3-32B	**2.08**	2.28	2.16	**1.38**	1.53	1.44	**0.51**	0.50	0.51	.63
DeepSeek-R1	**1.72**	2.47	2.00	**0.82**	1.64	1.12	**0.34**	0.51	0.41	.63
QwQ-32B	**1.74**	2.74	2.11	**0.86**	2.20	1.35	**0.32**	0.60	0.42	.63
DeepSeek-R1-Distill-Qwen-32B	**1.85**	2.07	1.95	**1.02**	1.21	1.10	**0.38**	0.41	0.40	.57
DeepSeek-R1-Distill-Qwen-7B	**1.51**	2.18	1.85	**0.60**	1.40	1.01	**0.27**	0.49	0.41	.49
Deepseek-R1-Distill-Qwen-1.5B	**1.41**	2.04	1.81	**0.53**	1.27	1.00	**0.27**	0.49	0.41	.36
Phi-4-Reasoning	**2.03**	2.83	2.53	**1.15**	2.10	1.75	**0.35**	0.58	0.49	.35
Instruction-tuned models without extended reasoning										
Qwen3-32B-NR	**1.66**	1.74	1.71	**0.70**	0.75	0.73	**0.29**	0.33	0.32	.34
Gemini-2.0-Flash	**1.43**	1.58	1.53	**0.43**	0.59	0.54	**0.20**	0.28	0.25	.34
Phi-4	**1.14**	1.51	1.41	**0.14**	0.51	0.41	**0.07**	0.23	0.19	.29
Qwen-2.5-32B	**1.33**	1.39	1.37	**0.37**	0.42	0.40	**0.16**	0.20	0.19	.28
GPT-4o	1.68	**1.40**	1.46	0.72	**0.40**	0.46	0.29	**0.19**	0.21	.19
Proprietary reasoning models with hidden traces										
Gemini-2.5-Flash	**1.56**	1.91	1.67	**0.59**	0.91	0.69	**0.28**	0.35	0.30	.67
GPT-o3-mini	**1.51**	1.65	1.59	**0.56**	0.70	0.63	**0.27**	0.32	0.30	.47
GPT-o1-mini	**1.36**	1.58	1.50	**0.36**	0.61	0.52	**0.19**	0.29	0.25	.37

Setup We use the 100-problem Omni-MATH subset used by ThinkARM (Gao et al., 2024; Li et al., 2025a), which contains CoT trajectories from 15 models. We train logistic regression classifiers with ℓ_1 or ℓ_2 regularization, selecting both the regularization type and hyperparameters by inner 5-fold cross-validation on AUROC. As baseline features, we include CoT length (Wu et al., 2025b), reasoning features, namely the count and proportion of reasoning tokens (e.g., tokens inside <think>...</think>). We also include self-revision features derived from markers such as "wait" and "aha" (Guo et al., 2025).[3] These self-revision features include raw count, token-level ratio, and binary presence. We also compare with ThinkARM (Li et al., 2025a), which represents each CoT by token ratios over eight episode labels.[4] For SHAPE, we use the analogous content-unit ratios over heuristic labels, i.e., the frequency distribution $u(h)$ defined in §2.3.

Results Table 1 reports the prediction results. CoT length alone performs near chance (0.504 ± 0.03), and adding the reasoning features does not improve performance (0.503 ± 0.03), indicating that surface-level token statistics carry little correctness signal. Self-revision markers achieve a notably high AUROC (0.618 ± 0.03) comparable to ThinkARM episode frequencies (0.618 ± 0.02), yet using far fewer features (three vs. eight). Notably, SHAPE frequency features achieve the best AUROC (0.664 ± 0.02), outperforming all baselines. This suggests that tracking heuristics provides stronger correctness signal than surface-level or episode-level representations.

3.2. Semantic-Space Metrics Reveal Structural Patterns in Reasoning

Beyond correctness prediction, we leverage SHAPE's semantic-space metrics to reveal the structural reasoning patterns of LLMs and how they organize their reasoning.

Setup We use the same dataset and models as §3.1[5]. For each model, we report three metrics over its CoT trajectories: the effective number of semantic spaces ($N^{\text{eff}}_{\text{space}}$), the effective number of semantic-space transitions ($N^{\text{eff}}_{\text{trans}}$), and transition ratio ($\rho = N^{\text{eff}}_{\text{trans}}/N^{\text{eff}}_{\text{space}}$), which measures the average number of transitions per semantic space.

Results As shown in Table 2, reasoning models consistently exhibit higher values across all semantic space metrics compared to instruction-tuned models: open-source

[3] Full marker list: wait, aha, hold on, recheck, re-check, reconsider, verify, to verify, let me verify, let's verify, check, let me check, let's check, confirm, let's confirm, alternatively, another way, another approach, no, no: but let's.

[4] Read, Analyze, Plan, Implement, Explore, Verify, Answer, and Monitor.

[5] Models comprise three groups: open-source reasoning models with full traces, instruction-tuned models without extended reasoning, and proprietary reasoning models with hidden traces. See Table 2 for the full list.

Table 3. CoT structural analysis under perturbation. Pass@1 is reported for original (O), simple (S), and hard (H) problems. D_{JS}^{freq} measures the Jensen–Shannon divergence of heuristic frequency distributions between the original and perturbed CoTs. $\Delta N_{\text{space}}^{\text{eff}}$ and $\Delta\rho$ report changes in the number of effective semantic spaces and transition ratio ($\rho = N_{\text{trans}}^{\text{eff}}/N_{\text{space}}^{\text{eff}}$), from the original to each perturbation. * denotes one-sided paired Wilcoxon signed-rank tests comparing hard and simple perturbations, with $p < .05$.

	Pass@1			D_{JS}^{freq}		$\Delta N_{\text{space}}^{\text{eff}}$		$\Delta\rho$	
Model	O	S	H	O→S	O→H	O→S	O→H	O→S	O→H
Qwen3-8B	.96	.94	.84	.21	.25*	-.02	.07*	-.00	.10*
Qwen3-32B	.92	.90	.76	.19	.24*	-.06	.08*	-.05	.11*
Nemotron-Cascade-8B	.92	.95	.71	.19	.24*	-.02	.14*	-.03	.08*
Olmo-3-7B-Think-RLVR	.97	.94	.76	.20	.25*	-.00	.07*	-.06	.03*

reasoning models show larger semantic space coverage ($N_{\text{space}}^{\text{eff}}$ of 1.81–2.53) and higher transition density (ρ of 0.40–0.51), while other models remain substantially lower on both axes ($N_{\text{space}}^{\text{eff}}$ of 1.37–1.71; ρ of 0.19–0.32). This suggests that extended reasoning does not merely elongate trajectories but induces qualitatively different traversal patterns over semantic space. A similar convergence of metrics appears when comparing correct and incorrect trajectories. Across most models, incorrect trajectories exhibit higher $N_{\text{space}}^{\text{eff}}$, $N_{\text{trans}}^{\text{eff}}$, and ρ than correct ones. Notably, incorrect trajectories exhibit higher ρ than correct ones across most models, indicating that they revisit the same semantic spaces more intensively rather than making forward progress, which may reflect the overthinking phenomenon (Wu et al., 2025b; Song et al., 2025).

4. Analysis on the Effects of Post-Training

Using validated SHAPE as an analysis tool, we now turn to two fundamental questions about post-training LLMs for mathematical reasoning: (1) can trained models deploy mathematically appropriate problem-solving strategies for problems that resemble those in the training data but require different strategies? and (2) how does RL post-training shape this strategic capability? For both questions, SHAPE reveals the underlying shifts in mathematical actions and interpretations that accuracy-level evaluation or surface-form similarity analyses could not capture.

Setup For this controlled setting, we use the 115-problem test split of MATH-Perturb (Huang et al., 2025), which pairs each *original* problem (level-5 MATH) with two perturbations that apply minimal textual modifications: a *hard perturbation* that alters the required solution approach, and a *simple perturbation* that preserves the solution method and structure. For each problem, we collect CoT traces on the original, simple, and hard versions from four post-trained models: Qwen3-8B, Qwen3-32B (Yang et al., 2025a), Nemotron-Cascade-8B (Wang et al., 2025), and Olmo-3-7B-Think-RLVR (Olmo et al., 2025). SHAPE annotation is performed using Grok-4.1-Fast. To distinguish whether models broaden their interpretations, or merely cycle harder within a fixed scope, we compute three pairwise measures between the original and perturbed problems' CoTs: D_{JS}^{freq} (change in heuristic frequency distribution u), $\Delta N_{\text{space}}^{\text{eff}}$ (change in the effective number of semantic spaces), and $\Delta\rho$ (change in transition ratio).

4.1. Perturbed Problems Induce Adaptive but Error-Like Reasoning

To examine whether models can deploy the mathematically appropriate problem-solving strategies demanded by a problem, we use a controlled setting where problems share a similar textual form but require a fundamentally different solution approach. This allows us to test whether models truly reorganize their mathematical interpretations, or merely adapt tactics within a fixed semantics. While prior work attributes performance drops on such perturbed problems to a reliance on solution strategies that models already handle well (Huang et al., 2025), models may recognize that the problem has changed and adapt both their heuristic choices and semantic-space dynamics, yet these adaptations fail to converge on a successful solution.

Results Table 3 shows substantial Pass@1 drops under hard perturbation across all four models. However, this is not mere repetition of memorized solutions; rather, models undergo structural changes in both local heuristic selections and global semantic-space dynamics when the underlying solution approach changes: D_{JS}^{freq}, $\Delta N_{\text{sp}}^{\text{eff}}$, and $\Delta\rho$ are all higher for hard than simple perturbations.[6] Yet this structural adaptation does not translate into successful reasoning. While models open more semantic spaces, they simultaneously revisit them far more often but failing to commit to successful interpretation.

4.2. Post-Training Narrows Heuristics Usage

We next investigate how RL post-training shapes the strategic capability of models. Specifically, we ask whether

[6]This is not a late-stage correction: heuristic distributions diverge within the first five and ten content units (Appendix E).

Table 4. Density and Coverage of post-trained model trajectories relative to base model trajectories in heuristic frequency space ($k = 3$, successful trajectories only, aggregated across all perturbation conditions). Density > 1 indicates that post-trained trajectories concentrate in the dense core of the base distribution; among such cases, lower Coverage indicates stronger mode-seeking. The cross-model baseline confirms that unrelated base models show neither high Density nor high Coverage.

Base	Post-trained	N_{base}	N_{PT}	Density	Coverage
Post-trained					
Qwen3-1.7B-Base	Qwen3-1.7B-GRPO	834	886	1.220	0.871
Olmo-3-7B	Olmo-3-7B-Think-RL-Zero	1229	1307	1.250	0.707
Olmo-3-7B	Olmo-3-7B-Think-RLVR	1507	1600	1.032	0.531
Cross-model baseline (unrelated distributions)					
Olmo-3-7B	Qwen3-1.7B-Base	71	66	0.520	0.437

RLVR post-training broadens or narrows reasoning strategies, and whether the remaining strategies after training are completely new or already present in the base model. Prior work suggests that RLVR redistributes probability mass over base model trajectories rather than introducing new ones (Yue et al., 2025; Wu et al., 2025a; Dang et al., 2025), but these findings rest on surface-form similarity or answer accuracy. We hypothesize that the same narrowing occurs at the level of heuristic choices and effort allocation: post-training concentrates successful trajectories into a narrower strategic region already present in the base model.

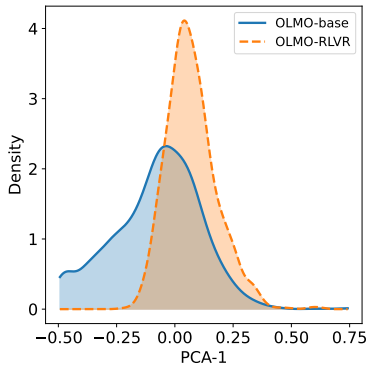

Figure 3. Projection of successful Olmo-3-{Base, Think-RLVR} trajectories onto the first principal component of heuristic frequency space. Olmo-3-Think-RLVR concentrates around the peak of the base distribution while leaving the left tail uncovered, suggesting that post-training narrows rather than reshapes the heuristic distribution.

Setup To measure heuristic overlap and concentration after post-training, we apply Density and Coverage (Naeem et al., 2020) to successful-trajectory heuristic frequency vectors $u(h)$, using the base as reference and the post-trained model as target ($k = 3$ nearest neighbors, cosine distance). Density measures concentration in the dense core of the base distribution, with values above 1.0 indicating stronger clustering around base trajectories. Coverage measures the fraction of base trajectories reached by the post-trained model: low values indicate that large portions of the base trajectories are left uncovered. We examine three pairs across two model families and post-training recipes: Olmo-3-7B with its Think-RLVR and Think-RL-Zero (Olmo et al., 2025), and Qwen3-1.7B-Base paired with our GRPO variant (Shao et al., 2024). We compare two unrelated base models as baselines (Olmo-3-7B and Qwen3-1.7B-Base). For each pair, we restrict to problems solved correctly by both models, aggregating up to five successful trajectories per problem across the MATH-Perturb test sets.

Results Results consistently show that post-training narrows heuristic space. Figure 3 shows that Olmo-3-7B-Think-RLVR concentrates around the dominant mode of the Olmo-3-7B base distribution while leaving the left tail uncovered. Table 4 shows a consistent pattern across all configurations: post-trained models concentrate reasoning into a narrower region of the strategic space defined by heuristic usage patterns. Every post-trained model exhibits Density above 1.0, indicating concentration in the dense core of the base distribution rather than spreading widely. Coverage is always below 1.0, meaning that some base-model heuristic patterns are not reproduced after post-training. The cross-model baseline—comparing two unrelated base models—shows both low Density (0.520) and low Coverage (0.437), showing that the high-Density and moderate-Coverage pattern is specific to base–post-trained pairs rather than generic overlap. In conclusion, these results provide a heuristic-level counterpart to prior findings that RL reduces the diversity of reasoning trajectories (Matsutani et al., 2025; Dang et al., 2025): post-training does not only reduce surface-form diversity, but also narrows the strategic space.

5. Heuristic-Augmented Reinforcement Learning

We further ask whether the diagnostic insights from SHAPE can guide a training improvement: *does simply considering*

Table 5. Preliminary results for Heuristic-Augmented GRPO on the MATH-Perturb test split. Plan-GRPO adds a planning step without heuristic information. HA-Plan-GRPO provides eleven mathematical heuristics during rollout and incorporates them into a planning-guided rollout. Bold indicates the best results.

Model	Original		Simple		Hard	
	Avg@64	Pass@64	Avg@64	Pass@64	Avg@64	Pass@64
Qwen3-1.7B-Base	23.54	77.40	23.10	79.13	11.84	57.39
+ Plan-GRPO	30.00	**80.00**	29.86	78.26	14.52	61.74
+ HA-Plan-GRPO (ours)	**36.80**	**80.00**	**35.80**	**79.13**	**17.72**	**62.61**

heuristic during training lead to a performance gain?

Setup To inject heuristic information during training, we train two variants of planning-based RLVR using Group Relative Policy Optimization (GRPO (Shao et al., 2024)). Plan-GRPO adds an explicit planning step during rollout following prior planning-based reasoning work (Jiao et al., 2024). HA-Plan-GRPO keeps the same planning structure but augments the rollout prompt with general mathematical heuristics. Both are trained on Qwen3-1.7B-Base with MATH (Hendrycks et al., 2021) training split and evaluated on the MATH-Perturb (Huang et al., 2025) test set. They share the same reward, verifier, optimizer, and evaluation prompt; only the training rollout prompt differs. Full training details including prompts are provided in Appendix F.

Results Table 5 reports performances across all splits. Both variants improve over the base model, confirming that the planning structure contributes to performance. However, importantly, HA-Plan-GRPO yields substantially higher both Avg@64 and Pass@64. Since the variants differ only in the rollout prompt, this suggests that heuristic information alone in the rollout prompt is sufficient to yield a measurable performance gain, demonstrating the potential of integrating heuristics into post-training.

6. Related Work

Chain-of-thought analysis. A growing body of work analyzes Chain-of-Thought (CoT) trajectories beyond final-answer accuracy, including reasoning length, self-revision markers, cognitive episode labels, and graph-based structure (Wu et al., 2025b; Guo et al., 2025; Li et al., 2025a; Marjanovic et al., 2026; Jiang et al., 2025; Xiong et al., 2025; Zhang et al., 2025). Other studies examine whether CoT steps are faithful to the model's internal computation, showing that verbalized reasoning may include post-hoc or decorative steps (Arcuschin et al., 2025; Lanham et al., 2023; Tanneru et al., 2024; Bogdan et al., 2025; Yang et al., 2025b). These works provide important tools for characterizing CoT behavior, but most operate over surface markers, generic reasoning episodes, or global structural patterns. SHAPE instead adopts concepts from mathematical problem-solving research—heuristics and semantic spaces—to track the mathematical interpretation under which each local action is taken (Pólya, 1945; Schoenfeld, 1985; Rott, 2014; Favier & Dorier, 2024).

Post-training for reasoning. Recent reasoning models rely heavily on reinforcement learning with verifiable rewards (RLVR), which improves mathematical reasoning through outcome-level correctness signals (Guo et al., 2025; Lambert et al., 2024; Yang et al., 2025a; Wang et al., 2025). However, recent work suggests that RLVR may improve sampling efficiency without expanding the model's reasoning repertoire, and may induce diversity collapse or mode-seeking during reasoning post-training (Yue et al., 2025; Zhao et al., 2025; Wu et al., 2025a; Huang et al., 2025; Liu et al., 2025). We complement this line by analyzing semantic-space and heuristic organization, rather than accuracy or output diversity.

7. Conclusion

We introduced SHAPE, a process-level framework for analyzing Chain-of-Thought reasoning through semantic spaces and heuristics from mathematical problem-solving research. Its heuristic features predict answer correctness more reliably than existing CoT representations, while its semantic-space metrics show that correct trajectories remain focused within fewer spaces. Diagnostically, SHAPE shows that models facing novel solution approaches change heuristics without expanding semantic-space scope, and that post-training concentrates successful trajectories in the dense core of the base model's heuristic distribution. Incorporating heuristic information into RLVR further improves performance across difficulty conditions. These findings suggest that process-level structure can provide useful diagnostic and training signals beyond final-answer accuracy. At the same time, SHAPE analyzes observable CoT traces, and its current validation is limited to mathematical benchmarks; extending the framework to other domains remains future work.

References

Arcuschin, I., Janiak, J., Krzyzanowski, R., Rajamanoharan, S., Nanda, N., and Conmy, A. Chain-of-thought reasoning in the wild is not always faithful. *arXiv preprint arXiv:2503.08679*, 2025.

Bogdan, P. C., Macar, U., Nanda, N., and Conmy, A. Thought anchors: Which llm reasoning steps matter? *arXiv preprint arXiv:2506.19143*, 2025.

Carlson, M. P. and Bloom, I. The cyclic nature of problem solving: An emergent multidimensional problem-solving framework. *Educational studies in Mathematics*, 58(1):45–75, 2005.

Dang, X., Baek, C., Kolter, J. Z., and Raghunathan, A. Assessing diversity collapse in reasoning. In *Scaling Self-Improving Foundation Models without Human Supervision*, 2025.

Ericsson, K. A. and Simon, H. A. Protocol analysis: Verbal reports as data cambridge. *EUA Massachusetts Institute of Technology*, 1993.

Favier, S. *Étude des processus de résolution de problèmes par essais et ajustements en classe de mathématiques à Genève*. PhD thesis, Université de Genève, 2022.

Favier, S. and Dorier, J.-L. Heuristics and semantic spaces for the analysis of students work in mathematical problem solving. *Educational Studies in Mathematics*, 115 (3):407–431, 2024.

Gandhi, K., Chakravarthy, A. K., Singh, A., Lile, N., and Goodman, N. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective STars. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=QGJ9ttXLTy.

Gao, B., Song, F., Yang, Z., Cai, Z., Miao, Y., Dong, Q., Li, L., Ma, C., Chen, L., Xu, R., et al. Omni-math: A universal olympiad level mathematic benchmark for large language models. *arXiv preprint arXiv:2410.07985*, 2024.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Huang, K., Guo, J., Li, Z., Ji, X., Ge, J., Li, W., Guo, Y., Cai, T., Yuan, H., Wang, R., et al. Math-perturb: Benchmarking llms' math reasoning abilities against hard perturbations. *arXiv preprint arXiv:2502.06453*, 2025.

Jaech, A., Kalai, A., Lerer, A., Richardson, A., El-Kishky, A., Low, A., Helyar, A., Madry, A., Beutel, A., Carney, A., et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Jiang, G., Liu, Y., Li, Z., Bi, W., Zhang, F., Song, L., Wei, Y., and Lian, D. What makes a good reasoning chain? uncovering structural patterns in long chain-of-thought reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025.

Jiao, F., Qin, C., Liu, Z., Chen, N., and Joty, S. Learning planning-based reasoning by trajectories collection and process reward synthesizing. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024.

Jost, L. Entropy and diversity. *Oikos*, 113(2):363–375, 2006.

Koichu, B., Berman, A., and Moore, M. Heuristic literacy development and its relation to mathematical achievements of middle school students. *Instructional Science*, 35(2):99–139, 2007.

Lambert, N., Morrison, J., Pyatkin, V., Huang, S., Ivison, H., Brahman, F., Miranda, L. J. V., Liu, A., Dziri, N., Lyu, S., et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024. URL https://arxiv.org/pdf/2411.15124.

Lanham, T., Chen, A., Radhakrishnan, A., Steiner, B., Denison, C., Hernandez, D., Li, D., Durmus, E., Hubinger, E., Kernion, J., et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023.

Li, M., Fan, C., Cheng, Y., Feizi, S., and Zhou, T. Schoenfeld's anatomy of mathematical reasoning by language models. *arXiv preprint arXiv:2512.19995*, 2025a.

Li, M., Zhang, N., Fan, C., Jiao, H., Fu, Y., Peters, S., Xu, Q., Lissitz, R., and Zhou, T. Understanding the thinking process of reasoning models: A perspective from schoenfelds episode theory. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025b.

Liu, M., Diao, S., Lu, X., Hu, J., Dong, X., Choi, Y., Kautz, J., and Dong, Y. ProRL: Prolonged reinforcement learning expands reasoning boundaries in large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=YPsJha5HXQ.

Marjanovic, S. V., Patel, A., Adlakha, V., Aghajohari, M., BehnamGhader, P., Bhatia, M., Khandelwal, A., Kraft, A., Krojer, B., Lù, X. H., Meade, N., Shin, D., Kazemnejad, A., Kamath, G., Mosbach, M., Stanczak, K., and Reddy, S. Deepseek-r1 thoughtology: Let's think about LLM reasoning. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=BZwKsiRnJI.

Matsutani, K., Takashiro, S., Minegishi, G., Kojima, T., Iwasawa, Y., and Matsuo, Y. Rl squeezes, sft expands: A comparative study of reasoning llms. *arXiv preprint arXiv:2509.21128*, 2025.

Naeem, M. F., Oh, S. J., Uh, Y., Choi, Y., and Yoo, J. Reliable fidelity and diversity metrics for generative models. In *International conference on machine learning*. PMLR, 2020.

Newell, A., Simon, H. A., et al. *Human problem solving*, volume 104. Prentice-hall Englewood Cliffs, NJ, 1972.

Olmo, T., Ettinger, A., Bertsch, A., Kuehl, B., Graham, D., Heineman, D., Groeneveld, D., Brahman, F., Timbers, F., Ivison, H., et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

Pólya, G. *How to Solve it: A New Aspect of Mathematical Method*. Princeton University Press, 1945. URL https://books.google.co.kr/books?id=YBhMnPXCy9YC.

Posamentier, A. S. and Krulik, S. *Problem-solving strategies for efficient and elegant solutions, grades 6-12: a resource for the mathematics teacher*. Corwin press, 2008.

Qwen. Qwen3.5: Towards Native Multimodal Agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Rott, B. Rethinking heuristics–characterizations and examples. In *Problem Solving in Mathematics Education–Proceedings of the 15th ProMath Conference*, pp. 176–192, 2014.

Schoenfeld, A. H. *Mathematical Problem Solving*. Academic Press, Orlando, FL, 1985. ISBN 9780126288704. doi: 10.1016/C2013-0-05012-8.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Sheng, G., Zhang, C., Ye, Z., Wu, X., Zhang, W., Zhang, R., Peng, Y., Lin, H., and Wu, C. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

Song, S., Oh, M., Lee, S., Jo, S., and Jo, Y. Thinkbrake: A simple test-time decoding control for efficient reasoning. *arXiv e-prints*, pp. arXiv–2510, 2025.

Su, J., Healey, J., Nakov, P., and Cardie, C. Between underthinking and overthinking: An empirical study of reasoning length and correctness in llms. *arXiv preprint arXiv:2505.00127*, 2025.

Tanneru, S. H., Ley, D., Agarwal, C., and Lakkaraju, H. On the hardness of faithful chain-of-thought reasoning in large language models. *arXiv preprint arXiv:2406.10625*, 2024.

Wang, B., Lee, C., Lee, N., Lin, S.-C., Dai, W., Chen, Y., Chen, Y., Yang, Z., Liu, Z., Shoeybi, M., et al. Nemotron-cascade: Scaling cascaded reinforcement learning for general-purpose reasoning models. *arXiv preprint arXiv:2512.13607*, 2025.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 2022.

Wu, F., Xuan, W., Lu, X., Liu, M., Dong, Y., Harchaoui, Z., and Choi, Y. The invisible leash: Why rlvr may or may not escape its origin. *arXiv preprint arXiv:2507.14843*, 2025a.

Wu, Y., Wang, Y., Ye, Z., Du, T., Jegelka, S., and Wang, Y. When more is less: Understanding chain-of-thought length in llms. *arXiv preprint arXiv:2502.07266*, 2025b.

xAI. Grok 4.1, November 2025. URL https://x.ai/news/grok-4-1.

Xiong, Z., Cai, Y., Li, Z., and Wang, Y. Mapping the minds of llms: A graph-based analysis of reasoning llms. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Yang, C., Si, Q., Duan, Y., Zhu, Z., Zhu, C., Li, Q., Chen, M., Lin, Z., and Wang, W. Dynamic early exit in reasoning models. *arXiv preprint arXiv:2504.15895*, 2025b.

Yue, Y., Chen, Z., Lu, R., Zhao, A., Wang, Z., Song, S., and Huang, G. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025.

Zhang, X. F., Mohananey, A., Chronopoulou, A., Papalampidi, P., Gupta, S., Munkhdalai, T., Wang, L., and Upadhyay, S. Do llms really need 10+ thoughts for "find the

time 1000 days later"? towards structural understanding of llm overthinking, 2025. URL https://arxiv.org/abs/2510.07880.

Zhao, R., Meterez, A., Kakade, S. M., Pehlevan, C., Jelassi, S., and Malach, E. Echo chamber: RL post-training amplifies behaviors learned in pretraining. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=dp4KWuSDzj.

A. Heuristic Taxonomy

In this section, we provide the full label set used for SHAPE annotation. The taxonomy is designed as an operational coding scheme for LLM-generated mathematical CoT trajectories. Its purpose is to give the annotator model a stable vocabulary for identifying the local mathematical action expressed in each content unit. The taxonomy is grounded in prior work on mathematical problem solving (Pólya, 1945; Koichu et al., 2007; Favier, 2022; Posamentier & Krulik, 2008). We consolidate these sources into 11 top-level heuristic families. These families cover representation changes, reinterpretation, formalization, problem classification, simplification, case decomposition, contradiction, analogy, pattern exploration, backward reasoning, and verification. Some families are further divided into sublabels when a finer distinction is useful for annotation. For example, H3 separates the introduction of notation from structural augmentation, and H11 separates direct checking, alternative derivation, backtracking, sanity checking, generalization, and reflection on rigor. Labels are assigned at the content-unit level. A unit receives a heuristic label when it performs a purposeful mathematical action that can shape the subsequent solution process. The labeling is multi-label: a single unit may both introduce a representation and check a constraint, or both explore a case and verify a claim. If a unit does not contribute a strategic mathematical action, it is assigned one of the non-heuristic labels N1–N4. This distinction is functional rather than lexical. A line containing equations may still be N2 if it only carries out routine computation after the strategy has already been fixed. Table 6 lists the heuristic labels and their source mappings. Table 7 lists the non-heuristic labels used to preserve non-strategic parts of the generated CoT trajectory.

Table 6. Taxonomy of Problem-Solving Strategies. This table summarizes the code, strategy name, detailed description, and theoretical sources associated with each heuristic. Here, P = Pólya (1945), K = Koichu et al. (2007), F = Favier (2022), and P&K = Posamentier & Krulik (2008).

Strategy Name	**Description & Sources**
H1. Changing the register of semiotic representation	This strategy involves translating the problem's representation from one semiotic register to another. It includes converting between natural language, algebraic, geometric, and visual representations to facilitate understanding or solving. **Sources:** Creating a model (K); Change the semiotic representation register (Changer de registre de représentation sémiotique) (F)
H2. Cognitive Reinterpretation	This involves changing the way an object or property in the problem is interpreted. It redefines the identity or attributes of an element in a way different from the initial presentation, without necessarily changing the register. **Sources:** Consider another way of interpreting the problem's objects (Envisager une autre façon d'interpréter les objets du problème) (F); Adopting a different point of view (P&K)
H3. Introduce Symbolic Representation, Formalization, and Structural Augmentation, including...	
H3a. Introduce Symbolic Representation and Formalization	The act of introducing new variables, labeling unknowns, or performing substitutions to make ambiguous targets operationally manageable. **Sources:** Notation; Setting up equations (P); Creating a model (K); Introduce names or notations (Introduire des noms ou des notations) (F)
H3b. Structural Augmentation	Constructing auxiliary objects, lemmas, or entirely new mathematical frameworks that are not present in the original problem. This is a creative addition to the problem space, such as drawing auxiliary lines, defining new functions, or shifting the problem into a new structural representation **Sources:** Auxiliary elements, Lemma (P); Introducing an auxiliary element (K); Introduce auxiliary elements (Introduire des éléments auxiliaires) (F)
H4. Problem Classification / Rephrase the Problem and Goal, including...	

H4a. Problem Categorization / Strategic Rephrasing of Goal / Breaking into Sub-goals	Explicitly stating the problem type, identifying applicable solution methods, or reformulating the main goal in clearer mathematical terms. **Sources:**What is the unknown? (P); Reformuler le problème (Reformulate the problem) (F)
H4b. Filtering Constraints	Strategically identifying the most essential constraints or conditions that guide the upcoming solution approach. **Sources:**Separate the various parts of the condition (P); Exploring a particular datum (K)
H5. Wishful Thinking (Simplify / Reduce the Problem and Conditions)	Temporarily modifying the problem to a simpler version to gain insight, verify formulas, or explore solution strategies. **Sources:** If you cannot solve the proposed problem (P); Reduce the problem to a simpler one (Réduire le problème à un problème plus simple) (F)
H6. Explicit Case Analysis, Decompose into Subproblems	Logically decomposing the problem into distinct cases, non-overlapping subsets, or sub-problems that, when combined, yield the full solution. The cases should ideally be exhaustive and mutually exclusive. **Sources:** Decomposing and recombining (P); Décomposer le domaine du problème et travailler cas par cas (Decompose the problem domain and work case by case) (F)
H7. Arguing by contradiction	A proof strategy where the negation of the proposition is assumed to derive a contradiction, thereby proving the original statement. **Sources:** Reductio ad absurdum and indirect proof (P); Arguing by contradiction (K)
H8. Analogy and Presenting Related Theorems, including...	
H8a. Analogy	Recalling previously solved problems, known methods, or applying a recently established logical procedure to a new target within the same problem. This involves recognizing structural similarities and transferring a strategy from one context (or one part of the equation) to another. **Sources:** Have you seen it before?; Do you know a related problem? (P); Activating a previous experience (K); Make a connection with a previously encountered problem (Faire le lien avec un problème déjà rencontré) (F); Solving a simpler analogous problem (P&K)
H8b. Presenting Related Theorems, Tools, or Properties	Introducing specific mathematical theorems, formulas, identities, or properties that are not provided in the problem statement but are necessary to advance the solution. **Sources:** Connect with a mathematical tool (theorem, property) (Faire le lien avec un outil mathématique (théorème, propriété)) (F)
H9. Experimental and Pattern Exploration, including...	
H9a. Exploring particular cases or numbers	Plugging in specific values, extreme/boundary values, or limits to discover patterns, build intuition, or verify feasibility. **Sources:** Specialization (P); Partial Induction (K); Explore a specific piece of data (Explorer une donnée particulière) (F); Finding a Pattern (P&K)
H9b. Exploration of symmetry	Identifying and exploiting mathematical or structural symmetry to reduce the solution space or simplify computation. **Sources:** Symmetry (P); Exploration of Symmetry (K); Exploit symmetry properties (Exploiter les propriétés de symétrie) (F)
H10. Thinking from the end to the beginning (Working backward)	Starting from the desired conclusion (target goal) and working logical steps backward to reach the known premises or to determine what would be sufficient to prove. **Sources:** Working backwards (P); Thinking Backward (K); Working backward (Travailler à reculons) (F); Working Backwards (P&K)
H11. Verification and Looking Back, including...	

H11a. Re-solving & Checking the Argument	Re-performing the same logical steps or calculations without a strategic change, or conducting a direct manual check of elements to verify a previous claim. **Sources:** Can you check the result? (P); Local Self-evaluating (K)
H11b. Deriving the Result Differently	Solving the same problem or sub-goal using a structurally different mathematical method to provide independent confirmation. **Sources:** Can you derive the result differently? (P); Local Self-evaluating (K)
H11c. Backtracking for self-verification	Realizing an error, finding a flaw in an assumption, or recognizing that the current approach is not working, and revising the direction. **Sources:** Backtracking (Retour arrière) (F)
H11d. Checking the Result / Sanity Check / Progress Review	Broadly covers any reflection on whether the solution is on the right track or checking feasibility. **Sources:** Can you check the result?; Test by dimension (P); Local Self-evaluating (K)
H11e. Generalization & Corollary	Extending the result to broader cases. Identifying general principles from specific solutions. **Sources:** Wisdom of proverbs (P); Generalization (K)
H11f. Reflect on Rigor & Wisdom	Evaluating the efficiency of the solution strategy, questioning the rigor, or meta-reflecting on definitions/rules. **Sources:** Why proofs? (P)

Table 7. Non-heuristic categories capturing reasoning steps that do not contribute strategic insight, including repetition, routine computation, irrelevant statements, and final answer reporting.

Name	Description
N1. Literal Repetition	Merely repeating the problem statement or giving a generic opening/closing without mathematical substance or added insight.
N2. Technical Performance	Routine calculations or simple algebraic manipulations that do not involve strategic planning or new insight; "doing the math" after the plan is set.
N3. Alien Statements	Unrelated thoughts or off-topic remarks.
N4. Answer	Stating the final answer.

B. Gold Heuristic Set Constructions

The gold set is intended as an adjudicated reference for validating the SHAPE annotation pipeline, not as an assumption that heuristic interpretation is observer-independent. Following prior work in mathematical problem-solving analysis, we treat annotation as an interpretive coding task whose reliability comes from explicit coding rules, examples, and consensus adjudication.

Trajectory selection. We constructed the gold set from six seed problems in MATH-Perturb, with problem IDs 12, 968, 1694, 1958, 2423, and 3156. For each seed problem, we used two versions: the original problem and its hard perturbation. We collected CoT trajectories for each version from four models: Qwen3-30B-A3B-Instruct, Qwen3-30B-A3B-Thinking, Qwen3-8B, and Nemotron-Cascade-8B. This yields $6 \times 2 \times 4 = 48$ CoT trajectories. This selection was designed to cover both original and structurally perturbed problem settings, as well as reasoning traces from models with different reasoning modes and model sizes.

Content-unit segmentation. We segmented each CoT trajectory into content units following the protocol-coding approach of Koichu et al. (Koichu et al., 2007). A content unit is the largest contiguous span of the trajectory that supports a particular heuristic interpretation. In practice, a content unit may consist of a few words, one sentence, or several sentences, depending on whether the span expresses a single problem-solving move. We introduced a new unit boundary when the model changed the local mathematical goal, shifted to a different action, moved from planning to execution, moved from computation to checking, or began a new attempt at representing or transforming the problem.

Table 8. Heuristic-tagging performance of candidate annotator models on the gold set. Weighted F1 reflects overall label agreement, while macro F1 emphasizes performance on rare heuristic classes.

Model	Weighted F1	Macro F1
Grok-4.1-Fast	**76.98**	**65.04**
GPT-5	72.25	63.55
GPT-5-mini	66.44	55.89
Gemini-3-Flash	62.83	41.79
Gemini-3-Flash-Lite	70.21	54.14
Qwen3.5-27B	70.44	61.36

Consensus coding. Four authors participated in the annotation process, including a graduate researcher in mathematics education. The trajectories were coded at the content-unit level. Disagreements were resolved through discussion, following the interpretive tradition in mathematical problem-solving research, where coding decisions are stabilized through comparison and adjudication rather than treated as purely mechanical labels. During discussion, annotators considered the problem statement, the local content unit, and the surrounding trajectory context. The final annotation was retained only after the annotators reached a shared interpretation of the mathematical role of the unit. Our procedure follows Koichu et al. (Koichu et al., 2007), who analyze thinking-aloud transcripts by segmenting them into content units and coding each unit according to its heuristic interpretation. In their protocol, content units are defined as the largest unbroken parts of the transcript that bear a particular heuristic interpretation, and disagreements are resolved by comparing coders' interpretations and adjudicating ambiguous cases. We adapt this procedure from human think-aloud data to LLM-generated CoT trajectories.

Borderline annotation example. One representative disagreement occurred in a trajectory generated by Qwen3-30B-A3B-Thinking on the hard perturbation of MATH-Perturb problem 1694. In this trajectory, the model tested its counting method on smaller ranges:

> "But just to make 100% sure, let's compute the total numbers with $\gcd(n, 28) = 2$ in a small range and see if our method works. Take $n > 2$, $n < 10$: numbers 3–9. ... So only $n = 6$ has $\gcd = 2$. Using our method ... Count=1. ... Another test: $n > 6$, $n < 20$."

Annotators initially differed on whether this unit should be treated as H9a, H11b, or H5. The H9a interpretation comes from the direct exploration of particular numerical cases. The H11b interpretation comes from the unit's role in the trajectory: the model has already derived a general counting method and uses these examples to check whether that method works. The H5 interpretation comes from reducing the original range to smaller test ranges. In adjudication, we retained all three labels because the unit performs all three roles: it verifies the solution by testing particular cases in a simplified version of the original problem.

C. Tagging Pipelines

Heuristic Tagging Model Selection We evaluate several candidate annotator models on the gold heuristic-tagging set. Table 8 reports weighted F1 and macro F1, where weighted F1 reflects overall agreement and macro F1 gives more weight to rare heuristic classes. Grok-4.1-Fast achieves the strongest overall performance, while Qwen3.5-27B also achieves comparable performance despite being an open-weight model available through HuggingFace.

Class-wise Agreement of the Open-Weight Annotator For reproducibility, we additionally report the class-wise performance of Qwen3.5-27B, our open-weight annotator. Table 9 reports precision, recall, F1, Cohen's kappa, and the number of gold and predicted instances for each heuristic class. The model obtains particularly strong agreement on frequent and visually salient classes such as H8, H11, and N, while performance is lower for rare classes such as H7 and H10, where the small number of gold instances makes the estimates less stable.

Semantic Space Tracking Model Prompts Calibration Semantic-space tracking requires a different validation strategy. Unlike heuristic tags, semantic spaces are trajectory-relative interpretations that evolve as the solver constructs and revises

Table 9. Class-wise heuristic-tagging performance of Qwen3.5-27B on the gold set. We report precision (P), recall (R), F1, Cohen's kappa, and the number of gold and predicted instances.

Class	F1	Kappa
H1	0.5272	0.4893
H2	0.4318	0.4176
H3	0.4884	0.4383
H4	0.6114	0.5049
H5	0.6250	0.6134
H6	0.6667	0.6626
H7	0.6667	0.6664
H8	0.7201	0.6252
H9	0.5934	0.5692
H10	0.5000	0.4988
H11	0.8115	0.6383
N	0.7216	0.6702

their representation of the problem. Because they are grounded in the solver's internal cognitive state rather than a closed label set, no objective gold standard exists against which to evaluate them directly. We therefore do not evaluate semantic-space tracking with the same label-level protocol used for heuristics. Instead, we implement it as a prompted state machine using the selected annotator model, and calibrate the prompt on the 48 annotated trajectories through iterative manual review.

D. Annotator Model Details

This appendix reports the algorithm and prompt skeletons used by the automated `SHAPE` annotation pipeline. To avoid duplicating long coding manuals in the paper, we omit the full guidebook text from the printed prompt listings. In the actual annotation runs, guidebook placeholders were expanded verbatim at runtime.

Specifically, `{HEURISTICS_GUIDEBOOK}` was replaced by the full heuristic annotation guidebook, and `{SEMANTIC_SPACE_GUIDE}` was replaced by the full semantic-space tracking guidebook. Both guidebooks, together with the prompt-building code and exact runtime templates, are included in the supplementary code release. Thus, each annotator call is defined by the prompt skeleton shown below plus the corresponding guidebook file in the supplementary materials.

For readability, the prompt listings below retain the guidebook placeholders rather than expanding them in full.

D.1. Content-Unit Segmentation Prompt

Content-Unit Segmentation Prompt

```
# Role
You are an expert researcher in mathematics education and cognitive psychology.
Your task is to segment a student's or AI model's Chain-of-Thought (CoT) transcript
into Content Units using the provided heuristic guidebook.

# Task Definition
Your primary goal is to segment the Chain-of-Thought transcript into Content Units
based on shifts in the problem-solving strategy.

A Content Unit is a single Plan-Execution Cycle:
A cohesive block of thought where the solver formulates a specific strategic
intent and immediately carries it out.

Segmentation Philosophy:
```

```
- Do not segment every sentence.
- Do not segment based only on pauses, filler words, or discourse markers.
- Segment only when the active strategy or underlying mathematical intent changes.
  For example, split when the solver switches from analyzing the problem to
  experimenting with numerical cases.

Crucial Segmentation Rules:
1. Coverage is mandatory: every sentence in the target range must be assigned to a
    unit.
2. Merge a plan and its immediate execution into one unit.
3. Use heuristic codes as justification for the segmentation boundary.
4. Start a new segment only when the active strategy changes.
5. Group consecutive sentences with the same strategic role into a single unit.
6. Merge weak or monitoring sentences such as "Okay", "Wait", or "Let me check"
   with neighboring substantive units.
7. Do not output N/A, empty codes, or non-heuristic labels.
8. Do not create single-sentence chunks for weak monitoring or transition sentences,
   especially when they are shorter than about 15 words.

# Reference: Heuristic Strategies
Use the heuristic taxonomy from the guidebook. The valid top-level heuristic
    families are:

H1:  Changing the register of semiotic representation
H2:  Cognitive reinterpretation
H3:  Introduce symbolic representation, formalization, and structural augmentation
H4:  Problem classification / rephrasing the problem and goal
H5:  Wishful thinking: simplifying or reducing the problem and conditions
H6:  Explicit case analysis / decomposing into subproblems
H7:  Arguing by contradiction
H8:  Analogy and presenting related theorems, tools, or properties
H9:  Experimental and pattern exploration
H10: Thinking from the end to the beginning / working backward
H11: Verification and looking back

# Critical Valid-Code Rule
The only valid codes in this segmentation stage are H1-H11.
Never use N1-N4, "Non-Heuristic", "Technical Performance", or any other non-H label.

# In-Context Example
Problem:
Find the least positive four-digit solution r of the congruence
r^2 + 4r + 4 = r^2 + 2r + 1 (mod 55).

Input CoT:
{SEGMENTATION_IN_CONTEXT_EXAMPLE_COT}

Output Segmentation:
{
  "chunks": [
    {"start_index": 1,  "end_index": 9,  "codes": ["H4"],        "reasoning": "..."},

    {"start_index": 10, "end_index": 15, "codes": ["H11","H8"],  "reasoning": "..."},

    {"start_index": 16, "end_index": 27, "codes": ["H8"],        "reasoning": "..."},

    {"start_index": 28, "end_index": 31, "codes": ["H11","H8"],  "reasoning": "..."},

    {"start_index": 32, "end_index": 46, "codes": ["H11"],       "reasoning": "..."},

    {"start_index": 47, "end_index": 57, "codes": ["H4"],        "reasoning": "..."}
  ]
}
```

```
# Instruction
Segment the following User Input, a Chain-of-Thought trace, into Content Units
based on shifts in the problem-solving strategy.

Important: Data minimization
- Output start_index and end_index only.
- Do not output the text content of the chunks.

# User Input
{INDEXED_SENTENCE_WINDOW}

Important:
1. You are provided with a window of sentences, each indexed as [1], [2], ...
2. Segment from index {TARGET_START_INDEX} up to roughly index {TARGET_END_INDEX}.
3. Start grouping from index {TARGET_START_INDEX}. Do not include earlier indices.
4. If a logical unit extends past {TARGET_END_INDEX}, stop before it and leave it
   for the next batch.
5. Group consecutive sentences with the same strategic role into a single chunk.
6. Never output a chunk containing only a weak monitoring sentence.
7. Every chunk must have at least one valid H1-H11 code.
8. Do not create single-sentence chunks for weak monitoring, confirmation, or
    transition.
9. Valid codes are H1-H11 only. Never use N1-N4 or any non-H prefix.
10. Output strictly as JSON with a "chunks" key:
    {
      "chunks": [
        {
          "start_index": 1,
          "end_index": 5,
          "codes": ["H4"],
          "reasoning": "Brief explanation."
        }
      ]
    }
11. Each item must include start_index, end_index, codes, and reasoning.
12. Keep reasoning concise: 1-2 sentences.
13. Do not quote long blocks of the input text.
14. Use single quotation marks inside reasoning if quotation is needed.
```

Runtime placeholders. `{INDEXED_SENTENCE_WINDOW}` is the sliding window of indexed sentences. `{TARGET_START_INDEX}` is the first sentence index to segment in the current batch. `{TARGET_END_INDEX}` is the approximate end index for the current batch. `{SEGMENTATION_IN_CONTEXT_EXAMPLE_COT}` is the fixed in-context example used for calibration.

D.2. Heuristic Tagging Prompt

Heuristic Tagging Prompt

```
In this project, we analyze the reasoning process of Large Reasoning Models (LRMs)
by identifying heuristic strategies used during mathematical problem solving.
Given one content unit from the model response, annotate it with heuristic or
non-heuristic codes from the guidebook.

The code set includes:
- H1-H11: mathematical heuristic strategies.
- N1-N4: non-heuristic categories, used only when no heuristic strategy is present.

Important Annotation Rules:
```

```
1. Use sub-codes when available. For categories with sub-codes, such as H4a/H4b,
   H9a/H9b, or H11a-H11f, use the specific sub-code if it applies. Use the parent
   code only as a fallback when no sub-code fits.
2. Multi-tagging is allowed. A content unit may receive multiple codes if multiple
   strategies are present.
3. Use non-heuristic codes only when no heuristic is present. If the unit contains
   even one H1-H11 action, output H codes only.
4. Cite evidence. For each code, quote the specific phrase or sentence from the
   current chunk that supports the annotation.

The [Guidebook] section provides the full definitions of the codes.
The [Math Problem] section provides the original math problem.
The [Previous Context] section provides recent chunks without previous tags.
The [Current Chunk] section provides the chunk to annotate.
The [Format] section specifies the required output format.

[Guidebook]
{HEURISTICS_GUIDEBOOK}
[End of Guidebook]

[Math Problem]
{PROBLEM_TEXT}
[End of Math Problem]

[Previous Context]
Recent chunks, text only:
Chunk {i-3}: {prev_chunk_text_i_minus_3_truncated_200chars}
Chunk {i-2}: {prev_chunk_text_i_minus_2_truncated_200chars}
Chunk {i-1}: {prev_chunk_text_i_minus_1_truncated_200chars}
Chunk {i}:   {prev_chunk_text_i_truncated_200chars}
[End of Previous Context]

[Current Chunk]
Chunk to annotate:
{CURRENT_CHUNK_TEXT}
[End of Current Chunk]

[Format]
Output JSON exactly in the following structure:
{
  "annotations": [
    {
      "code": "H4",
      "evidence": "Direct quote from the chunk serving as evidence.",
      "reasoning": "Brief explanation of why this code applies."
    }
  ]
}

The "annotations" field must be a list.
Each object must contain exactly one code.

# Heuristic-First Decision Rule
Before annotating, follow this decision tree:
1. First, scan carefully for any heuristic activity, H1-H11.
2. If you find even one heuristic, tag with H codes only.
3. Only if there is no heuristic activity, use N codes.

Common mistakes to avoid:
- Do not tag modular arithmetic operations as N2 when they introduce or use a
  mathematical representation or tool. Use H1 or H8 when appropriate.
- Do not tag finding an inverse or invoking a theorem as N2. Use H8 when it
  functions as a mathematical tool.
```

```
- Do not tag strategic substitutions, such as "Let u = x^2", as N2. Use H3.
- Do not tag problem-to-equation conversion as N2. Use H1 or H3.

N2 is only for routine arithmetic or algebra after the strategy has already been set,

such as "2+3=5" or "2x+3x=5x".

Now annotate the current chunk using the heuristic-first approach.
```

Runtime placeholders. `{HEURISTICS_GUIDEBOOK}` is the full heuristic guidebook. `{PROBLEM_TEXT}` is the original math problem. `{prev_chunk_text_*}` are up to four recent prior chunks, truncated to 200 characters. If no previous chunks exist, this field is replaced with "There are no previous chunks." `{CURRENT_CHUNK_TEXT}` is the content unit being annotated.

D.3. Semantic-Space Tracking Algorithm

Algorithm 1 Semantic Space Tracking

Require: Content units $u_1, \ldots, u_T$, space memory buffer $\mathcal{M}$, semantic space tracking model $\mathcal{A}$
Ensure: Semantic space ID sequence $s_1, \ldots, s_T$

```
Initialize current space s_cur ← 1, M ← {1}
for each content unit u_t do
    if heuristic(u_t) is representation-changing        // H1, H2, H3, H5, H8, H11 then
        d ← A(u_t, M)
        if d = NEW then
            s_cur ← |M| + 1, update M
        else if d = RETURN then
            s_cur ← j                                   // j ∈ M
        else
            (MAINTAIN)
            Stay in current space
        end if
    end if
    Assign s_t ← s_cur
end for
```

D.4. Semantic-Space Tracking Prompt

Semantic-Space Tracking Prompt

```
You are an expert mathematical cognitive scientist analyzing the reasoning traces
of a Large Reasoning Model.

Your task is to track state changes in the Semantic Space: the problem's
fundamental representation format, constraints, and overarching mathematical
    framework.

This prompt is called only when the current content unit contains at least one
representation-changing heuristic. The trigger set is:
H1, H2, H3, H5, H8, H11.

# Critical Rule for NEW vs RETURN
Before creating a NEW space, strictly compare the current chunk against the
[Memory of Past Spaces]. If the model is reverting to a previously established
equation, constraint, representation, or tool that exists in memory, even from
```

```
many steps earlier, output RETURN with the corresponding target_space_id.

Create a NEW space only when the current chunk introduces a fundamentally
unprecedented mathematical environment.

# Critical Rule for RETURN vs MAINTAIN
If the model explores and abandons a hypothetical sub-space within the same chunk
and then goes back to the currently active space, output MAINTAIN, not RETURN.

RETURN should be used only when the active space from the previous chunk is
different from the space being resumed.

# Critical Rule for H11b and Alternative Methods
For alternative derivations or checking steps:
- Output NEW only if the alternative introduces a fundamentally different
  mathematical structural framework, such as switching from coordinate-based
  algebra to a coordinate-free Gram matrix approach.
- Output MAINTAIN if the alternative uses a different formula or minor variation
  within the same framework.
- Look for a deep structural shift, not merely a local change in technique.

[Guidebook]
{SEMANTIC_SPACE_GUIDE}
[End of Guidebook]

[Math Problem]
{PROBLEM_TEXT}
[End of Math Problem]

[Memory of Past Spaces]
Memory of Past Spaces:

[ID: 0]
  Register: Natural Language Context
  Constraints: Original problem constraints
  Core Tools: None
  Summary: Initial mathematical problem phrasing.
  Anchor Text: The original problem instruction.

[ID: {space_id}]
  Register: {register}
  Constraints: {constraints}
  Core Tools: {core_tools}
  Summary: {summary}
  Anchor Text: {anchor_text}

... one block per previously created space ...
[End of Memory of Past Spaces]

[Recent Context]
Recent Context, last 10 chunks:
Chunk {i-9}: {prev_chunk_text_truncated_200chars}
...
Chunk {i-1}: {prev_chunk_text_truncated_200chars}
[End of Recent Context]

[Current Chunk]
Current Chunk to Assess:
{CURRENT_CHUNK_TEXT}

Triggered Heuristics:
{COMMA_SEPARATED_H_TAGS}
[End of Current Chunk]
```

```
[Format Instruction]
Output JSON exactly in the following format.
Do not use Markdown block syntax around the JSON.
Reply only with the raw JSON object.

For NEW, include a new_space_definition.
For RETURN or MAINTAIN, do not include new_space_definition.

When creating a NEW space definition, provide:
- register: the main representation format or mathematical language.
- constraints: the constraints active in this semantic space.
- core_tools: the main mathematical tools, objects, or operations.
- summary: a brief 1-2 sentence description of the strategy and intent.
- anchor_text: the exact equation, constraint declaration, or pivotal quote from
  the chunk that defines this space.

{
  "decision": "NEW",
  "rationale": "Explanation of why this decision was made according to the guidebook
    .",
  "target_space_id": 1,
  "new_space_definition": {
    "register": "",
    "constraints": "",
    "core_tools": "",
    "summary": "",
    "anchor_text": ""
  }
}

For RETURN, use this form:
{
  "decision": "RETURN",
  "rationale": "Explanation of why the current chunk resumes a previous space.",
  "target_space_id": 1
}

For MAINTAIN, use this form:
{
  "decision": "MAINTAIN",
  "rationale": "Explanation of why the current chunk remains in the active space.",
  "target_space_id": 1
}
```

Runtime placeholders. `{SEMANTIC_SPACE_GUIDE}` is the semantic-space tracking guidebook. `{PROBLEM_TEXT}` is the original math problem. `{space_id, register, constraints, core_tools, summary, anchor_text}` are fields from the accumulated semantic-space memory. `{prev_chunk_text_*}` are up to ten recent prior chunks, truncated to 200 characters. `{CURRENT_CHUNK_TEXT}` is the current content unit. `{COMMA_SEPARATED_H_TAGS}` is the list of heuristic tags assigned to the current content unit.

E. Early-Stage Heuristic Divergence Under Perturbation

Section 4.1 shows that hard perturbations lead models to change their heuristic selections more than simple perturbations. A possible concern is that this effect may arise only later in the trajectory, after the model has already drifted from its initial plan or become trapped in repeated revisitation. To test this, we recompute D_{JS}^{freq} after truncating each CoT to its first k content units, with $k \in \{5, 10\}$. For each original–perturbed pair, we construct the heuristic frequency distribution $u(h)$ from the truncated prefix and compute the Jensen–Shannon divergence between the original and perturbed trajectories.

Table 10 reports the prefix-level results. Hard perturbations produce larger heuristic-frequency divergence than simple

Table 10. Early-stage heuristic-frequency divergence under perturbation (post-trained models only). D_{JS}^{freq} is computed from heuristic frequency distributions after truncating each CoT to the first 5 or first 10 content units. Values compare the original CoT with the simple (O→S) or hard (O→H) perturbed CoT. Significance markers denote one-sided paired tests (hard > simple): $^*p < .05$.

	D_{JS}^{freq} (first 5 units)		D_{JS}^{freq} (first 10 units)	
Model	O→S	O→H	O→S	O→H
Qwen3-8B	.16	.19*	.10	.14*
Qwen3-32B	.15	.20*	.09	.13*
Nemotron-Cascade-8B	.17	.21*	.10	.14*
Olmo-3-7B-Think-RLVR	.17	.21*	.10	.13*

perturbations for every post-trained model, both in the first 5 units and in the first 10 units. All hard-versus-simple comparisons pass a one-sided paired test at $p < .05$. For the first 5 units, D_{JS}^{freq} under hard perturbation ranges from .19 to .21, compared with .15 to .17 under simple perturbation. The same pattern holds for the first 10 units: hard perturbations yield .13–.14, while simple perturbations remain around .09–.10.

These results show that the heuristic shift observed in Section 4.1 is not only a late-stage artifact. Models begin to alter their mathematical actions near the start of the solution when the perturbation changes the required approach. However, as shown in Table 3, this early tactical adjustment does not translate into a broader semantic-space expansion. The models change what they do, but they still tend to reason within a similar semantic-space scope.

F. Experiment Details

This section provides experimental details for the experiments described in §4.2, and §5. We use 2×NVIDIA B200 GPUs for GRPO training.

F.1. Prompts

We use the following planning prompt templates for all GRPO training. The placeholder `{problem}` is replaced with problems in the training dataset.

Planning Prompt Template

`{problem}`

Solve the problem step by step using explicit planning.

Before solving, write a short plan that states the goal and the sub-goals or intermediate results you expect to need. Then carry out the plan one step at a time. The plan should guide the next action, not merely describe it afterward.

Format:

[Plan] Goal: Sub-goals:

[Solution]

[Step 1] Reasoning:

[Step 2] Reasoning:

...

[Final Answer] Please reason step by step, and put your final answer within \boxed{}

Heuristic-Augmented Planning Prompt Template

`{problem}`

Solve the problem step by step using heuristic-guided planning.

At each major reasoning step, first choose one mathematical heuristic, briefly explain why it is useful, and then perform the corresponding reasoning move. The heuristic should guide the next action, not merely describe it afterward.

Use only heuristics that genuinely help solve the problem. The list is not a checklist; do not force irrelevant heuristics into the solution.

Available heuristics: 1. Change the Representation
2. Reinterpret the Object
3. Introduce New Symbols or Structures
4. Restructure the Goal
5. Wishful Thinking: Simplify Temporarily
6. Divide into Cases
7. Argue by Contradiction
8. Draw on Mathematical Knowledge
9. Explore with Examples and Symmetry
10. Work Backward from the Goal
11. Monitor, Verify, and Look Back

Format:

[Plan] Goal: Potentially useful heuristics:

[Solution]

[Step 1] Heuristic: Why this heuristic applies: Reasoning:

[Step 2] Heuristic: Why this heuristic applies: Reasoning:

...

[Final Answer] Please reason step by step, and put your final answer within \boxed{}

F.2. Training Details

Table 11 summarizes the training hyperparameters. We train for 200 steps on the MATH training splits for approximately 3–3.5 hours. We set the maximum generation length to 2048 for training efficiency. GRPO algorithm is implemented in `verl` (Sheng et al., 2025)

Table 11. Hyperparameters used for GRPO training.

Parameter	**Value**
Base Model	Qwen/Qwen3-1.7B-Base
Training Batch Size	32
Rollouts per Prompt	4
Number of GPUs	2×B200
Optimizer	AdamW
Temperature	1.0
Top-p	1.0
Top-k	-1
Max Response Length	2048
Learning Rate	1×10^{-6}
Warmup Steps	10
Training Steps	200