Schoenfeld's Anatomy of Mathematical Reasoning by Language Models

Ming Li*, Chenrui Fan*, Yize Cheng*, Soheil Feizi, Tianyi Zhou
University of Maryland, College Park
{minglii,cfan42,yzcheng,sfeizi}@umd.edu, tianyi.david.zhou@gmail.com
Project: https://github.com/MingLiiii/ThinkARM

Abstract

Large language models increasingly expose reasoning traces, yet their underlying cognitive structure and steps remain difficult to identify and analyze beyond surface-level statistics. We adopt Schoenfeld's Episode Theory as an inductive, intermediate-scale lens and introduce **ThinkARM** (***A**natomy of **R**easoning in **M**odels*), a scalable framework that **explicitly abstracts reasoning traces into functional reasoning steps** such as *Analysis*, *Explore*, *Implement*, *Verify*, etc. When applied to mathematical problem solving by diverse models, this abstraction reveals reproducible thinking dynamics and structural differences between reasoning and non-reasoning models, which are not apparent from token-level views. We further present two diagnostic case studies showing that exploration functions as a critical branching step associated with correctness, and that efficiency-oriented methods selectively suppress evaluative feedback steps rather than uniformly shortening responses. Together, our results demonstrate that episode-level representations make reasoning steps explicit, enabling systematic analysis of how reasoning is structured, stabilized, and altered in modern language models.

1 Introduction

Large language models (LLMs) have demonstrated remarkable performance on complex reasoning tasks (OpenAI, 2024c; Marjanović et al., 2025; Comanici et al., 2025; Qwen Team, 2025a), particularly when generating explicit chains of thought (CoT) (Wei et al., 2023). Consequently, evaluation of reasoning models has predominantly focused on outcome-oriented metrics such as accuracy, solution length, or aggregate token counts (Lightman et al., 2023; Jiang et al., 2025a). While these measures are effective for comparing final performance, they provide limited insight into how these models organize their reasoning traces and how different reasoning behaviors emerge across models.

It is still unclear which parts of a generated chain-of-thought correspond to problem understanding, exploration, execution, or verification, making it difficult to interpret model behavior beyond surface statistics. This opacity is particularly salient when studying "overthinking" (Chen et al., 2025b; Fan et al., 2025; Kumar et al., 2025), where longer or more elaborate reasoning does not necessarily translate into improved correctness (Feng et al., 2025b), yet the underlying thinking dynamics and structure are hard to characterize systematically. Various reasoning behaviors that are often discussed *conceptually*, e.g., abstract planning versus concrete execution, open-ended exploration versus evaluative checking, lack rigorous formulation and quantitative comparison across models.

To better interpret such reasoning traces, a natural question is whether they contain meaningful structure at an intermediate level of abstraction beyond individual tokens. Motivated by prior work (Li et al., 2025c) that introduced Schoenfeld's Episode Theory (Schoenfeld, 1985) as a framework for characterizing problem-solving behaviors, we adopt episode-level representations as an ***inductive lens*** for analyzing LLM reasoning traces. Episode Theory conceptualizes problem solving in terms of functional episodes, thereby providing an interpretable intermediate-scale representation that bridges low-level token statistics and high-level reasoning intents. A condensed example is shown in Figure 1.

While earlier work (Li et al., 2025c) first leveraged this theory to reasoning models, their scope is limited to one model and one dataset. Thus, it remains unclear whether such an episode-level abstraction can reveal systematic, reproducible structure in LLM reasoning at scale. In this work, we build upon this foundation and extend episode-level annotation to a broader, comparative setting, using it to examine what principal patterns emerge when diverse reasoning traces are viewed through

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 32773–32802

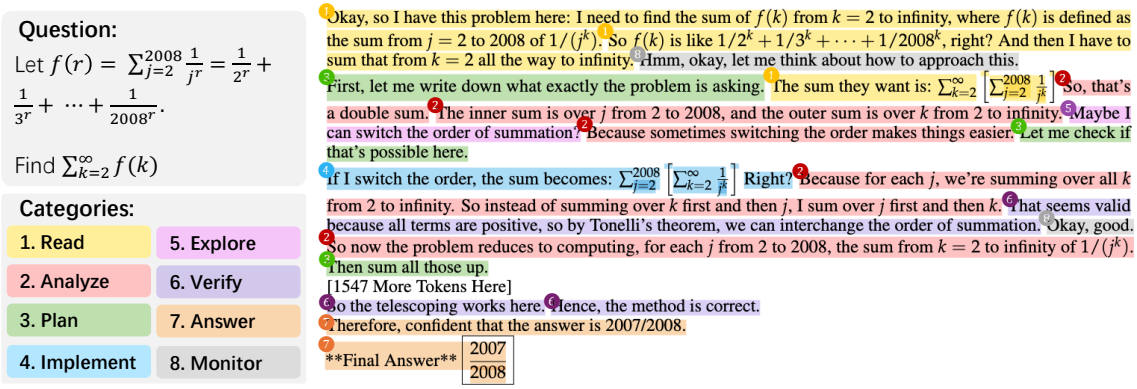

Figure 1: A condensed example of a reasoning trace that is annotated in our framework. Each sentence in response is tagged with one of the eight episode categories.

a theory-grounded abstraction. Our study is organized around three research questions. **RQ1:** *Do reasoning traces exhibit consistent linguistic and dynamic structure when viewed through episode-level representations?* **RQ2:** *How do reasoning dynamics differ across models and reasoning styles, as indicated by episode sequencing and transition patterns?* **RQ3:** *Can we utilize this framework to analyze the correlation of thinking patterns to final correctness, and the structural differences between overthinking and efficient models?*

To address these questions, we apply **ThinkARM** (*Anatomy of Reasoning in Models*), an episode-level annotation framework grounded in Schoenfeld's Episode Theory (Harskamp and Suhre, 2007), to a large-scale analysis of mathematical reasoning traces from a diverse set of LLMs. Concretely, we curate a corpus of $410,991$ sentences generated by 15 models solving 100 problems from a subset of Omni-MATH (Gao et al., 2024). To support reliable automated analysis, we construct a human-verified gold set of $7,067$ sentences and evaluate multiple state-of-the-art LLMs as automatic annotators, and finally select GPT-5 (OpenAI, 2025b) for full-scale episode labeling due to its strongest agreement with human annotations. This pipeline enables consistent, sentence-level episode annotation at scale and sets up a controlled setting for comparing reasoning dynamics across models and methods.

Contributions. We extend a cognitive science-inspired episode annotation framework to an automatic, scalable, sentence-level representation that supports large-scale analysis of reasoning traces and conduct a systematic study of reasoning dynamics across a diverse set of LLMs. Moreover, we demonstrate the practical utility of episode-level representations through downstream case studies on correctness and efficiency, illustrating how reasoning dynamics can be analyzed beyond outcome-based metrics.

Key Findings.

1. When reasoning traces are analyzed at the episode level, ***a functional progression from abstract reasoning to concrete execution, and finally to evaluative control, consistently emerges.*** Episodes associated with analysis and exploration use more abstract, conceptual language and ***decrease*** steadily as reasoning progresses, while execution-oriented episodes ***dominate*** the middle of the trace through sustained concrete operations. In contrast, verification-related episodes are characterized by evaluative and meta-level language and ***increase*** toward the end of the reasoning process.

2. Comparing reasoning and non-reasoning models, the difference is not merely how many tokens they generate, but how reasoning is structured. ***Non-reasoning models allocate most of their response trace to execution***, with episode transitions largely following a feed-forward pattern toward implementation. In contrast, ***reasoning models distribute effort across analysis, exploration, execution, and verification, and exhibit frequent iterative Explore-Monitor/Verify loops.***

3. Through our correctness-oriented case study, we find that ***exploration reflects uncertainty and serves as a critical branching point***: correct solutions more often route exploration into monitoring or re-analysis, whereas incorrect solutions tend to continue execution or terminate prematurely after exploration.

4. Through our efficiency-oriented case study, we find that ***different efficient reasoning methods selectively suppress evaluation-oriented episodes and feedback loops, leading to varying degrees of divergence*** from the reasoning patterns of the base model. Episode-level analysis thus reveals which episodes can be removed to gain efficiency, beyond token-level pruning.

Together, these findings make explicit a range of intermediate-scale reasoning behaviors ***that are often discussed intuitively but rarely characterized structurally.***

2 The ThinkARM Framework

In this section, we formalize our analytical methodology. We first ground our approach in cognitive science theory in mathematical problem solving and then detail the implementation of ThinkARM, a scalable, automated pipeline to quantify the reasoning dynamics of LLMs with high precision.

2.1 Schoenfeld's Episode Theory

To scientifically dissect the reasoning traces of LLMs, we ground our framework in Schoenfeld's Episode Theory (Schoenfeld, 1985), a seminal framework in cognitive science originally developed to decode the black box of human mathematical problem-solving. Schoenfeld's theory is descriptive, derived from the rigorous analysis of hundreds of hours of videotaped "think-aloud" protocols. By observing students and mathematicians tackling mathematical problems, they identified that the performance is not distinguished by superior domain knowledge alone, but by the dynamic regulation of that knowledge. The theory frames problem-solving as a temporally ordered sequence of "episodes" that reveal the solver's evolving goal structure and metacognitive decisions[1].

Schoenfeld's framework ultimately consists of 7 episodes: the original 6 episodes, including *Read*, *Analyze*, *Plan*, *Implement*, *Explore*, and *Verify*, plus a later-added *Monitor* episode that specifically captures the solver's metacognitive behaviors. Episode theory has since become a foundational analytic lens in mathematics-education research and in studies of human reasoning, offering a fine-grained vocabulary for tracing cognitive control and strategy shifts (Harskamp and Suhre, 2007). Li et al. (2025c) first applied this theory to annotate reasoning traces of LLMs with thinking behaviors. However, their work is limited to single-task scenarios and single-model case studies and does not provide a systematic analysis of the reasoning dynamics of diverse LLMs.

[1]More discussion on mathematical problem solving in cognitive science can be found in Appendix A

	Reasoning		Non-Reasoning		Overall	
Model	Acc	Kappa	Acc	Kappa	Acc	Kappa
GPT-4.1	85.75	82.39	89.34	**85.36**	86.10	82.74
GPT-5	**86.02**	**82.54**	**89.34**	85.35	**86.33**	**82.85**
Gemini-2.5-Flash	82.45	78.21	87.16	82.35	82.90	78.67
Gemini-2.5-Pro	80.53	75.60	84.43	78.62	80.89	75.96

Table 1: Performance of candidate annotators on the human-annotated gold set. GPT-5 shows the highest agreement with human annotations overall and is used for further large-scale annotation.

2.2 A Refined Taxonomy

While Schoenfeld's original taxonomy captures key functional stages of human problem solving, it does not include an explicit state for structural convergence. In the context of LLM reasoning, where models are trained to produce a final answer in a prescribed format, this distinction becomes practically important. We therefore extend the taxonomy by introducing an *Answer* episode, resulting in ***Eight*** episodes, which allows us to explicitly identify when the model commits to producing a final solution and to analyze convergence behavior separately from verification or monitoring, as shown in Figure 1.

Prior work (Li et al., 2025c) explored episode-based annotation using hierarchical schemes that combine paragraph-level and sentence-level labels. In this study, we adopt sentence-level annotation only, as it provides a uniform granularity that facilitates large-scale aggregation, transition analysis, and comparison across models. This design choice reflects a trade-off toward scalability and analytical simplicity, rather than a claim about the relative merits of different annotation granularities.

2.3 Data Collection and Gold Annotation

To construct a diverse and representative dataset for our analysis, we sample problems from Omni-MATH using its domain annotations. Specifically, we stratify the full benchmark by domain and select problems from each group proportionally, aiming to preserve domain coverage while maintaining diversity in problem types and difficulty. This procedure yields a subset of 100 problems spanning a broad range of mathematical topics.

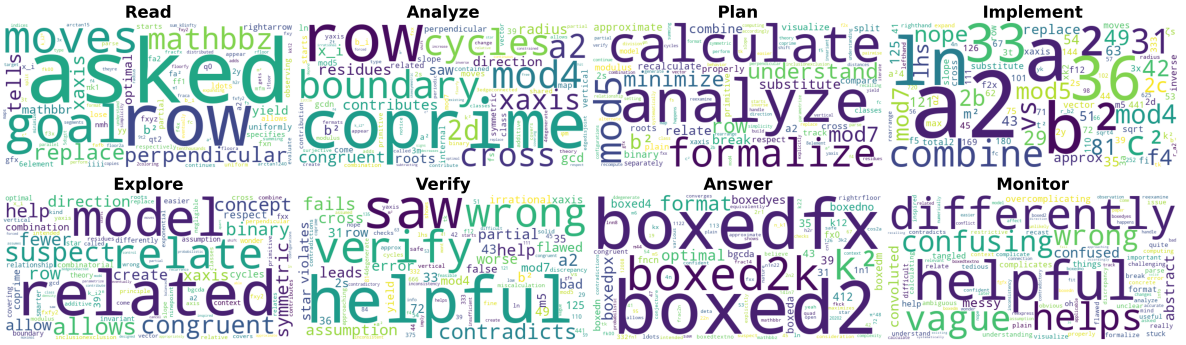

Figure 2: Word clouds visualizing the most frequent semantic tokens for each cognitive episode. **The distinct lexical distributions highlight the semantically separable cognitive patterns captured by ThinkARM.**

We then collect reasoning traces utilizing 15 widely used LLMs[2] to solve each problem, resulting in $1,500$ responses and a total of $410,991$ sentences. The collected traces include both thinking and answering tokens for native reasoning models and their distilled variants, and only answering tokens for standard instruction-following baselines and proprietary reasoning models without accessible thinking tokens. This diverse model coverage enables comparative analysis across model families and reasoning paradigms.

To support the reliable evaluation of automated episode annotation, we construct a human-verified gold set. From the 100 sampled problems, we select 9 representative problems following the same domain-based grouping strategy and manually annotate[3] all resulting reasoning traces using the refined episode taxonomy and guidebook. This process produces $7,067$ annotated sentences, which we use as the gold standard for testing the effectiveness of the automatic annotators and selecting the annotation model used in our large-scale analysis.

2.4 The ThinkARM Framework

In ThinkARM, we utilize LLMs for automatic annotation. During the automatic annotation process, we provide the detailed annotation guidebook, which consists of the definition and examples of each episode, along with the previous contexts and annotated categories. Furthermore, when annotating each sentence, we not only ask models to generate the annotated category, but also generate the justifications for each annotation to ensure the reliability of the annotation[4].

For the annotation model selection, we evaluate the performance of several state-of-the-art models, including GPT-4.1 (OpenAI, 2025a), GPT-5 (OpenAI, 2025b), Gemini-2.5-Flash (Comanici et al., 2025), and Gemini-2.5-Pro, on the gold standard annotated traces. The performance of the annotation models is shown in Table 1, containing the accuracy and kappa scores on the traces from reasoning and non-reasoning models. Based on the performance, we select GPT-5 for the full-scale annotation and utilize its results for further analysis.

3 Episode Patterns of Reasoning Models

With ThinkARM, we now move beyond surface-level performance metrics to empirically analyze the cognitive dynamics of current LLMs.

3.1 Episode Divergence

Our analysis asks whether different functional modes of reasoning, often discussed intuitively, manifest as separable patterns in language. Figure 2 visualizes the most frequent tokens associated with each episode and shows that episodes occupy distinct lexical regions, suggesting that **ThinkARM captures meaningful behavioral differences rather than superficial variation.**

***Analyze* vs. *Implement*:** A clear separation emerges between abstract reasoning and concrete execution. Language associated with *Analyze* emphasizes conceptual and structural elements of the problem (e.g., "coprime", "boundary"), reflecting the construction and manipulation of an abstract problem representation. In contrast, *Implement* is dominated by procedural and symbol-level language (e.g., variable names and concrete values), indicating step-by-step execution of a chosen approach. Thus, ThinkARM captures a genuine shift from conceptual thinking to solid implementation in reasoning behavior.

***Verify* vs. *Monitor*:** Two qualitatively different forms of reflection also emerge. *Verify* is character-

[2]The complete list of models is in Appendix B

[3]The detailed annotation guidebook is in Appendix G

[4]The detailed ThinkARM Framework is in Appendix F and the annotation prompt is in Appendix H

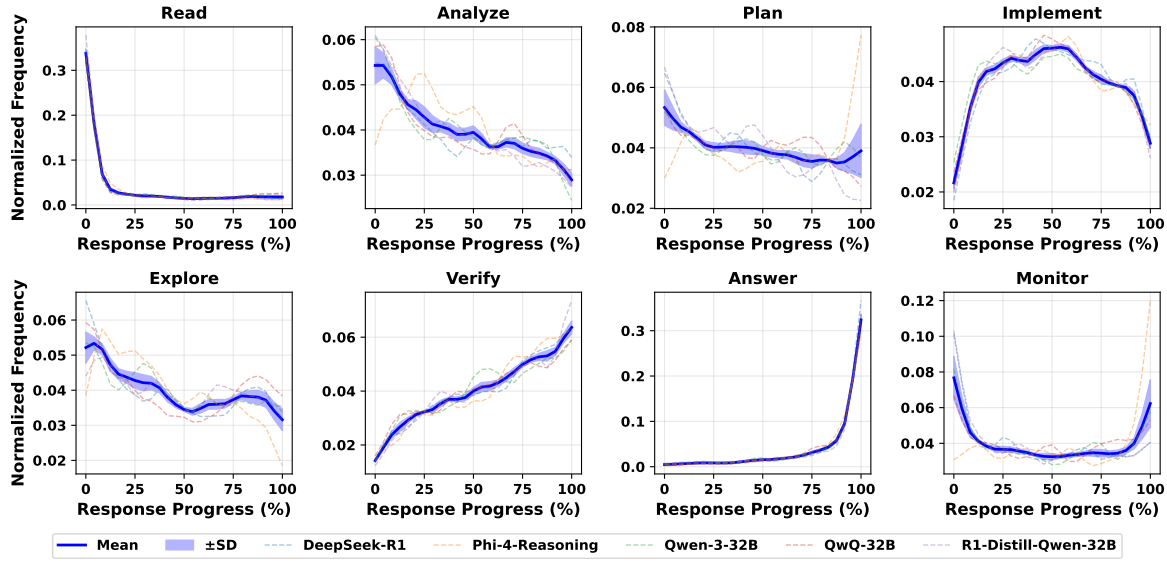

Figure 3: Thinking dynamics of cognitive episodes **reveal a three-phase "heartbeat" pattern of reasoning models:** (1) Initialization, dominated by *Read*, *Analyze*, and *Plan*; (2) Execution, where *Implement* peaks; and (3) Convergence, characterized by a surge in *Verify* and *Monitor* before the final *Answer*.

ized by decisive evaluative language (e.g., "wrong", "helpful"), indicating explicit checking of logical correctness or validity. *Monitor*, by contrast, is associated with meta-level expressions of uncertainty or progress tracking (e.g., "confusing", "messy"), reflecting awareness of the state of the reasoning process rather than evaluation of a specific step. These two behaviors that are separated lexically support treating verification and monitoring as distinct reasoning modes, which are important in later temporal and structural analyses.

***Plan* vs. *Explore*:** Forward-looking reasoning also exhibits two separable modes. *Plan* is dominated by directive verbs and goal-oriented language (e.g., "calculate", "formalize"), signaling commitment to a concrete strategy. In contrast, *Explore* is marked by tentative and hypothesis-driven expressions (e.g., "suspect", "maybe"), reflecting open-ended search over possible solution paths. This separation highlights exploration as a distinct behavioral mode characterized by uncertainty, rather than merely an early stage of execution.

3.2 Temporal Dynamics

By normalizing each reasoning trace to a 0-100% progress scale[5], we analyze how episode frequencies evolve over the generation (Figure 3). A consistent coarse-grained temporal organization emerges, captured by ThinkARM, across reasoning models. We refer to this pattern as the *cognitive heartbeat* of machine reasoning.

Early-stage scaffolding: Episodes associated with initial scaffolding (*Read*, *Analyze*, *Plan*, *Explore*) exhibit distinct decay patterns. *Read* drops sharply after the beginning, while *Analyze* and *Plan* decay more gradually, indicating that structural reasoning persists beyond the first few steps rather than being confined to a fixed planning prefix. *Explore* is typically front-loaded as well, consistent with early hypothesis search that narrows as execution proceeds.

Mid-stage execution: *Implement* exhibits a characteristic bell shape trajectory, peaking in the middle of the trace. This pattern suggests that concrete execution occupies the longest continuous region of the reasoning process, providing the backbone onto which other behaviors (e.g., exploration and verification) attach. The model shifts from high-level strategizing to mechanical execution (symbolic manipulation and calculation) as the core engine of the response.

Late-stage convergence: *Verify* increases steadily over progress, and *Monitor* follows a U-shaped profile with elevated mass near the beginning and end. Together, these trends indicate that evaluative and process-regulatory behaviors become more prominent as generation approaches completion, rather than appearing only as a final end check. Finally, *Answer* remains near-zero for most of the trace and rises sharply near the end, reflecting that answer commitment is concentrated in the terminal stage.

[5]Details in Appendix B

Model	Percentage (%)								Token Count (#)								Avg
	Read	Ana	Plan	Imp	Exp	Ver	Ans	Mon	Read	Ana	Plan	Imp	Exp	Ver	Ans	Mon	Tok
DeepSeek-R1	3.47	30.75	5.67	36.35	9.21	10.16	2.86	1.54	321	2844	524	3363	852	940	265	142	9250
R1-Distill-Qwen-1.5B	4.18	26.93	4.20	34.39	13.23	11.43	3.80	1.84	414	2666	416	3404	1309	1131	376	182	9897
R1-Distill-Qwen-7B	3.80	25.82	4.44	36.47	12.24	11.97	3.40	1.86	336	2280	392	3220	1081	1057	300	165	8831
R1-Distill-Qwen-32B	3.12	25.23	4.75	36.49	10.94	10.67	6.85	1.95	300	2422	456	3502	1050	1025	658	187	9598
QwQ-32B	3.03	24.53	6.28	35.71	13.98	11.05	3.32	2.09	378	3068	785	4466	1748	1382	416	261	12504
Phi-4-Reasoning	4.16	27.83	6.42	37.51	11.56	8.91	2.75	0.87	415	2771	639	3734	1151	887	273	86	9957
Qwen-3-32B (R)	2.88	25.58	7.79	36.86	11.56	10.54	3.04	1.76	372	3308	1007	4767	1495	1363	393	227	12931
GPT-4o	8.49	22.66	12.96	40.77	2.58	4.84	6.94	0.77	59	156	89	281	18	33	48	5	690
Gemini-2.0-Flash	5.63	19.24	4.33	61.17	0.97	4.45	4.14	0.08	63	215	48	682	11	50	46	1	1115
Qwen-2.5-32B	5.85	22.78	13.58	45.62	0.88	3.39	7.25	0.65	41	160	95	320	6	24	51	5	700
Phi-4	4.87	15.66	6.66	64.12	0.21	4.27	3.89	0.31	64	206	88	844	3	56	51	4	1316
Qwen-3-32B	6.87	13.03	6.82	65.76	0.97	3.42	2.82	0.30	182	345	181	1742	26	91	75	8	2649
Gemini-2.5-Flash	2.64	28.04	6.47	52.28	0.42	6.75	3.05	0.35	60	642	148	1197	10	155	70	8	2290
GPT-o1-mini	9.42	22.97	8.64	42.01	0.71	4.26	11.32	0.67	53	129	49	237	4	24	64	4	563
GPT-o3-mini	6.17	31.87	8.04	35.34	0.97	4.66	9.90	3.04	63	328	83	363	10	48	102	31	1027

Table 2: Episode-level allocation across model families. **Standard instruction-following models are strongly *Implement*-heavy, whereas reasoning models exhibit a more balanced allocation with substantial mass on *Analyze*, *Explore*, and *Verify*. Distilled models largely preserve the teacher's allocation profile across scales.**

3.3 Episode Coverage

Table 2 reports the episode-level allocation of generated tokens. We group models into: (i) open-source reasoning models where full reasoning traces are available, (ii) standard instruction-following models evaluated on their direct responses, and (iii) proprietary reasoning models where only the final responses are observable.

***From Doing to Thinking*:** A clear allocation gap emerges between reasoning and non-reasoning models. Standard instruction-following models allocate the majority of tokens to *Implement*, with minimal mass assigned to *Explore*, *Verify*, or *Monitor*. In contrast, reasoning models exhibit a more balanced profile, allocating substantially more budget to *Analyze* and *Explore*, and maintaining non-trivial allocation to *Verify*. **This suggests that the distinguishing factor is not merely response length, but how the generation budget is distributed across reasoning behaviors.**

***The Nature of Non-reasoning Responses*:** For proprietary reasoning models where only the final formulated answers are accessible, the episode allocation from the observable outputs is much closer to the non-reasoning group than to open-trace reasoning models. In other words, the externally visible responses exhibit limited allocation to exploration- and verification-associated episodes.

***Distillation Preserves Structure*:** Within the distilled series, we observe that episode allocation profiles remain similar across model sizes. R1-Distill-Qwen-1.5B exhibits an episode distribution close to its teacher DeepSeek-R1 despite large differences in parameter count. This indicates that distillation can transfer not only answers but also episode-level reasoning structure, as reflected in allocation patterns.

R vs. Reasoning (Answer)			R vs. Non-Reasoning Models		
Idx	Score	Pattern	Idx	Score	Pattern
1	0.2862	Exp-Mon	1	0.2510	Exp-Mon
2	0.2815	Ver-Exp	2	0.2405	Mon-Exp
3	0.2771	Mon-Exp	3	0.2073	Exp-Ana-Imp
4	0.2754	Exp-Plan	4	0.2033	Ana-Exp-Ana
5	0.2605	Exp-Ver	5	0.2028	Exp-Ana-Exp
6	0.2579	Exp-Imp	6	0.1974	Exp-Ana
7	0.2568	Exp-Ana-Exp	7	0.1917	Ver-Exp
8	0.2567	Exp-Plan-Imp	8	0.1838	Exp-Ver
9	0.2560	Imp-Ver-Exp	9	0.1830	Exp-Mon-Exp
10	0.2519	Ana-Exp	10	0.1806	Ana-Exp-Mon

Table 3: Top discriminative episode N-grams ranked by Mutual Information (MI). Left: patterns that distinguish a reasoning model's reasoning trace from the final answer segment. Right: patterns that distinguish reasoning traces from non-reasoning model responses. **Higher MI indicates a stronger association between the pattern and the source group.**

3.4 Transition Patterns

Beyond marginal episode allocations, episode *transitions* characterize how reasoning behaviors interact with each other. We convert each trace into a symbolic episode sequence (e.g., *Read* $\rightarrow$ *Plan* $\rightarrow$ *Implement*) and use an information-theoretic criterion to identify distinctive local structures[6]. Concretely, we compute the Mutual Information (MI) between the presence of episode N-grams and the source group, and extract the most discriminative patterns. Formally, the MI between an episode N-gram pattern presence X and the source group variable G is defined as:

$$I(X;G) = \sum_{x\in\{0,1\}} \sum_{g\in\mathcal{G}} p(x,g) \log \frac{p(x,g)}{p(x)\,p(g)}, \qquad (1)$$

[6]Details in Appendix B

Positive Correlation			Negative Correlation		
#	Feature	β	#	Feature	β
1	Trans (Exp → Mon)	+0.41	1	Ratio (Exp)	-0.54
2	Trans (Exp → Ana)	+0.31	2	Trans (Exp → Ver)	-0.45
3	Trans (Mon → Ana)	+0.28	3	Trans (Exp → Ans)	-0.41
4	Trans (Read → Ver)	+0.27	4	Count (Total)	-0.36
5	Ratio (Think)	+0.22	5	Trans (Imp → Read)	-0.32
6	Trans (Ans → Ver)	+0.22	6	Trans (Imp → Imp)	-0.28
7	Ratio (Ver)	+0.21	7	Trans (Ana → Mon)	-0.26
8	Trans (Read → Mon)	+0.18	8	Trans (Ver → Read)	-0.25
9	Trans (Read → Read)	+0.17	9	Trans (Ver → Plan)	-0.25
10	Ratio (Mon)	+0.16	10	Trans (Mon → Read)	-0.22

Table 4: Top predictive episode-level features for correctness in a diagnostic Lasso logistic regression case study. Features are ranked by their coefficient magnitude (β), where the sign and magnitude of β indicate the direction and relative strength of association with correctness under the fitted model.

where $x \in \{0, 1\}$ indicates whether a given episode N-gram is present in a trace, and g denotes the source group. Since MI is symmetric and only measures the strength of association, we further use conditional probability to determine which group each top-k discriminative pattern is attributed to.

Reasoning vs. Answer Tokens: Comparing the reasoning portion of a reasoning model to the final formulated answer tokens, we find that the most discriminative patterns are short feedback loops involving *Explore*, *Monitor*, and *Verify* (Table 3). In particular, *Exp-Mon* and *Mon-Exp* rank among the top patterns, indicating frequent alternation between exploration and process monitoring during the reasoning trace. Patterns such as *Ver-Exp* further suggest that verification is often followed by renewed exploration rather than immediate convergence. In contrast, the final answer tokens are characterized by more feed-forward transitions with limited recurrence of these evaluative loops.

Reasoning vs. Non-Reasoning Models: When comparing reasoning traces to responses from standard instruction-following models, we observe an overlapping set of discriminative patterns: non-reasoning responses are dominated by feed-forward transitions, where exploration-monitoring-verification loops are much less prevalent. This indicates that the gap between **reasoning and non-reasoning models is reflected not only in how much time is allocated to different behaviors, but also in the presence of recurrent transitions that interleave exploration with evaluation.**

4 Applications

4.1 How Episodes Correlate to Correctness

To demonstrate the utility of our framework, we conduct a correctness-oriented case study examining how episode-level patterns are associated with solution correctness. Using a stratified sample of 500 reasoning traces from the 5 representative open-source reasoning models, we formulate a binary classification setting that predicts answer correctness from quantitative cognitive features extracted from episode annotations[7].

Methodology We map each reasoning trace to a feature vector consisting of three components: (i) global statistics (total token count, thinking-token count, and thinking-token ratio), (ii) episode intensities (token ratios for each of the eight episodes), and (iii) transition features (the flattened 8×8 episode transition matrix capturing frequencies of state shifts). Then, we fit a Lasso-regularized logistic regression model to predict the correctness of a reasoning trace:

$$\mathcal{L} = -\sum_i \left[y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) \right] + \lambda \|\mathbf{w}\|_1, \quad (2)$$

where $\hat{y} = \sigma(\mathbf{w}^\top \mathbf{x})$, $\sigma(\cdot)$ denotes the sigmoid function and the ℓ_1 penalty encourages sparsity in the coefficient vector. Table 4 reports the top features ranked by coefficient magnitude (β). Here, each coefficient β corresponds to a weight in $\mathbf{w}$ for a specific episode-level feature. A positive (negative) β indicates that higher values of the feature are associated with increased (decreased) likelihood of a correct answer under this model, and the magnitude of β reflects its relative importance among the selected features.

Results The most predictive positive features highlight how exploratory behavior is resolved during successful reasoning. In particular, *Explore* → *Monitor* and *Explore* → *Analyze* rank among the strongest positive coefficients, suggesting that correct solutions tend to route exploratory uncertainty into meta-level monitoring and renewed conceptual analysis. Similarly, *Monitor* → *Analyze* indicates that monitoring is often followed by additional analysis rather than immediate execution, consistent with an uncertainty-to-reasoning redirection pattern. Verification-related transitions also appear as informative signals at different points in

[7] Details in Appendix E

Model	Percentage (%)								Token Count (#)								Avg
	Read	Ana	Plan	Imp	Exp	Ver	Ans	Mon	Read	Ana	Plan	Imp	Exp	Ver	Ans	Mon	Tok
R1-Distill-Qwen-1.5B	4.18	26.93	4.20	34.39	13.23	11.43	3.80	1.84	414	2666	416	3404	1309	1131	376	182	9897
L1 (1.5B) (Aggarwal and Welleck, 2025a)	9.33	18.34	6.01	34.03	15.14	6.99	8.52	1.64	227	447	147	829	369	170	208	40	2437
ThinkPrune (1.5B) (Hou et al., 2025)	6.48	20.75	5.16	39.39	10.74	8.37	7.31	1.80	263	841	209	1597	436	339	297	73	4054
A&Z (Arora and Zanette, 2025)	5.12	28.32	4.69	36.90	8.76	9.94	4.63	1.64	272	1505	249	1960	465	528	246	87	5312

Table 5: Cognitive behavior divergence of different efficient reasoning methods. L1 and ThinkPrune drastically reduce the budget for *Verify* and *Analyze*, while the last one maintains a distribution closer to the baseline.

R vs. L1			R vs. ThinkPrune			R vs. A&Z		
Idx	Score	Pattern	Idx	Score	Pattern	Idx	Score	Pattern
1	0.3762	N-V-N	1	0.1506	N-V-N	1	0.1039	M-E
2	0.2491	V-N-V	2	0.1465	V-N-I	2	0.0928	E-N-I
3	0.2476	M-N-V	3	0.1240	V-N	3	0.0919	M-E-I
4	0.2291	V-N	4	0.1200	I-N-V	4	0.0911	I-N-M
5	0.2261	M-V	5	0.1120	E-V-N	5	0.0897	N-M-I
6	0.2233	N-M	6	0.1073	V-N-V	6	0.0836	E-V-I
7	0.2144	I-V-M	7	0.1070	V-N-E	7	0.0789	V-A-M
8	0.2075	V-M	8	0.1018	I-V-N	8	0.0746	I-M-N
9	0.2017	I-N-M	9	0.0924	N-I-V	9	0.0701	E-I
10	0.1949	V-N-I	10	0.0881	V-M-P	10	0.0692	I-N-V

Table 6: Cognitive patterns that are lost in efficiency models compared to the baseline. L1 shows high distinctive scores, indicating a significant loss of complex verification loops (*V-N-V*). Conversely, A&Z shows much lower divergence scores, suggesting it preserves the baseline's topological structure more effectively.

the trace. *Read* → *Verify* and *Answer* → *Verify* suggest that traces associated with correctness more frequently include explicit checking both near the beginning and near the end.

On the negative side, a higher *Explore* ratio is a strong risk indicator, suggesting that sustained exploration without subsequent stabilization is associated with incorrect outcomes. Two additional failure-associated patterns emerge. First, *Explore* → *Verify* reflects verification applied before exploration has been consolidated into a coherent hypothesis. Second, *Implement* → *Read* indicates breakdowns during execution that coincide with reverting to problem re-reading, a signal of disrupted reasoning flow.

This study illustrates **how episode-level representations can surface interpretable transition-level signatures associated with correctness**, complementing outcome-based evaluation with a structured diagnostic view of reasoning behavior.

Further Discussion More importantly, these findings suggest that reasoning performance may be improved by explicitly shaping such structural patterns during training or inference. For instance, prior work has shown that encouraging cognitive behaviors such as backtracking and self-reflection can lead to more robust reasoning trajectories (Kim et al., 2025; Yang et al., 2025). Similarly, incorporating key cognitive patterns into training data has been found to significantly enhance model performance (Gandhi et al., 2025). Beyond training-time interventions, recent studies demonstrate that reasoning behaviors can also be steered at inference time via activation control, providing a practical pathway to translate diagnostic insights into performance gains (Venhoff et al., 2025). Taken together, these results highlight a promising direction: leveraging episode-level diagnostics not only for analysis, but also as a foundation for systematically improving reasoning processes.

4.2 Episode Divergence for Efficient Models

Efficient reasoning methods for LLMs are often evaluated primarily through surface metrics such as response length, leaving it unclear which reasoning behaviors are being altered. In this case study, we use ThinkARM to characterize how different efficiency paradigms reshape episode-level dynamics. Specifically, we compare a baseline model (R1-Distill-Qwen-1.5B) against three representative strategies: (1) L1 (Aggarwal and Welleck, 2025a), (2) ThinkPrune (Hou et al., 2025), and (3) the model proposed by Arora and Zanette (2025), referred as A&Z.

Table 5 summarizes how episode-level token allocation changes for these methods. Compared to the baseline, L1 and ThinkPrune substantially reduce the budget assigned to evaluative and conceptual episodes (e.g., *Verify* and *Analyze*) and shift the profile toward a more *Implement*-heavy allocation. In contrast, A&Z maintains an allocation profile closer to the baseline, retaining a substantial *Verify* and *Analyze* budget. These patterns suggest that efficiency methods are not uniformly compressive: they can induce qualitatively different redistributions of reasoning behaviors.

To further localize which transition-level structures are most affected, Table 6 reports the most distinctive episode N-grams suppressed by each efficiency strategy relative to the baseline by computing the Mutual Information. L1 exhibits high divergence scores (peaking at 0.37), partic-

Model	Read	Ana	Plan	Imp	Exp	Ver	Ans	Mon
DeepSeek-R1	21.82	13.04	4.13	29.69	6.88	8.19	14.88	1.38
Qwen-3-32B (R)	6.53	5.70	5.40	16.48	3.52	55.89	4.18	2.30
Gemini-2.0-Flash	14.81	9.04	3.85	55.58	0.00	1.35	14.62	0.77
Qwen-3-32B	21.94	2.14	22.91	37.67	0.00	0.00	13.40	1.94

Table 7: Episode-level token allocation (%) on a 50-problem subset of GSM8K. Rows are separated into reasoning (top) and non-reasoning (bottom) models. Compared to Omni-Math (Table 2), all models shift more budget to *Read* and *Answer* on easier problems, while non-reasoning models largely suppress *Explore* and *Verify*.

ularly for loop-like patterns such as *N-V-N* (*Analyze→Verify→Analyze*) and *V-N-V*, indicating that recurrent verification loops are strongly attenuated. ThinkPrune shows a similar but generally weaker suppression of such loop structures. By contrast, A&Z yields much lower divergence scores (peaking around 0.10), suggesting that it preserves more of the baseline transition topology while still reducing overall cost.

Overall, this case study illustrates that efficiency is not behaviorally neutral: **strategies that achieve similar surface-level reductions in length can correspond to markedly different changes in episode-level allocation and transition structure.**

5 Results on Problems with Lower Difficulty Level

The analyses presented so far focus on Omni-Math, an olympiad-level benchmark that stresses reasoning at its limits. To examine whether the observed episode-level patterns are specific to this high-difficulty regime or reflect more general properties of model behavior, we apply ThinkARM to a 50-problem subset of GSM8K (Cobbe et al., 2021), a widely used benchmark of grade-school math word problems. We annotate four models: two reasoning models (DeepSeek-R1 and Qwen-3-32B (R)) and two non-reasoning models (Gemini-2.0-Flash and Qwen-3-32B). Table 7 reports the episode-level allocation. Several patterns emerge when comparing these results to the Omni-Math allocations in Table 2.

Easier problems shift budget toward reading and answering. All four models allocate substantially more tokens to *Read* on GSM8K than on Omni-Math, consistent with simpler problems requiring more time to process relative to the total response. *Answer* proportions are also higher, reflecting shorter reasoning paths before commitment.

Non-reasoning models largely suppress exploration and verification. On Omni-Math, non-reasoning models already show limited *Explore* and *Verify* allocations; on GSM8K this suppression is near-total. Both Gemini-2.0-Flash and Qwen-3-32B assign essentially zero budget to *Explore* and *Verify*, suggesting that these models skip reflective checking entirely on problems they perceive as routine.

Model-specific traits become more pronounced. Cross-difficulty differences reveal characteristic model signatures. The reasoning mode of Qwen-3-32B allocates 55.89% of its GSM8K budget to *Verify*—far higher than on Omni-Math—suggesting it applies proportionally more checking effort on problems where execution is inexpensive. Gemini-2.0-Flash concentrates 55.58% on *Implement*, consistent with a direct-execution strategy that is amplified when problems offer little structural resistance.

Overall, these results show that the main qualitative distinctions between reasoning and non-reasoning models identified on Omni-Math are preserved, and in several respects sharpened, on medium-difficulty problems, supporting the generality of the episode-level framework across difficulty regimes.

6 Conclusion

In this work, we introduce ThinkARM as an inductive, intermediate-scale framework that abstracts reasoning traces into functional reasoning steps grounded in cognitive theory. Through large-scale empirical analysis, we show that this abstraction makes previously opaque reasoning structures explicit, revealing consistent temporal organization and interpretable diagnostic patterns related to correctness and efficiency. Our framework provides a principled lens for analyzing, comparing, and diagnosing reasoning behavior in modern language models.

Limitations

Large-scale episode annotation relies on an automatic annotator, which may introduce labeling noise despite strong agreement with human annotations on a verified gold set. Moreover, our experiments focus primarily on mathematical problem solving; extending episode-level analysis to other domains and reasoning settings remains an important direction for future work.

References

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

Pranjal Aggarwal and Sean Welleck. 2025a. L1: Controlling how long a reasoning model thinks with reinforcement learning. *Preprint*, arXiv:2503.04697.

Pranjal Aggarwal and Sean Welleck. 2025b. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*.

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 225–237, St. Julian's, Malta. Association for Computational Linguistics.

Daman Arora and Andrea Zanette. 2025. Training language models to reason efficiently. *arXiv preprint arXiv:2502.04463*.

Maciej Besta, Julia Barth, Eric Schreiber, Ales Kubicek, Afonso Catarino, Robert Gerstenberger, Piotr Nyczyk, Patrick Iff, Yueling Li, Sam Houliston, Tomasz Sternal, Marcin Copik, Grzegorz Kwaśniewski, Jürgen Müller, Łukasz Flis, Hannes Eberhard, Hubert Niewiadomski, and Torsten Hoefler. 2025. Reasoning language models: A blueprint. *Preprint*, arXiv:2501.11223.

Paul C. Bogdan, Uzay Macar, Neel Nanda, and Arthur Conmy. 2025. Thought anchors: Which llm reasoning steps matter? *Preprint*, arXiv:2506.19143.

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

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

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

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, Luke Marris, Sam Petulla, Colin Gaffney, and et al. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *Preprint*, arXiv:2507.06261.

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

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, and etc. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Chenrui Fan, Ming Li, Lichao Sun, and Tianyi Zhou. 2025. Missing premise exacerbates overthinking: Are reasoning models losing critical thinking skill? *arXiv preprint arXiv:2504.06514*.

Gongfan Fang, Xinyin Ma, and Xinchao Wang. 2025. Thinkless: Llm learns when to think. *Preprint*, arXiv:2505.13379.

Sicheng Feng, Gongfan Fang, Xinyin Ma, and Xinchao Wang. 2025a. Efficient reasoning models: A survey. *Preprint*, arXiv:2504.10903.

Yunzhen Feng, Julia Kempe, Cheng Zhang, Parag Jain, and Anthony Hartshorn. 2025b. What characterizes effective reasoning? revisiting length, review, and structure of cot. *Preprint*, arXiv:2509.19284.

Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D. Goodman. 2025. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *Preprint*, arXiv:2503.01307.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, and 1 others. 2024. Omni-math: A universal olympiad level mathematic benchmark for large language models. *arXiv preprint arXiv:2410.07985*.

Carole Greenes. 1995. Mathematics learning and knowing: A cognitive process. *Journal of Education*, 177(1):85–106.

E. Harskamp and C. Suhre. 2007. Schoenfeld's problem solving theory in a student controlled learning environment. *Computers & Education*, 49(3):822–839.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. 2025. Thinkprune: Pruning long chain-of-thought

of llms via reinforcement learning. *Preprint*, arXiv:2504.01296.

Dongzhi Jiang, Renrui Zhang, Ziyu Guo, Yanwei Li, Yu Qi, Xinyan Chen, Liuhui Wang, Jianhan Jin, Claire Guo, Shen Yan, Bo Zhang, Chaoyou Fu, Peng Gao, and Hongsheng Li. 2025a. Mme-cot: Benchmarking chain-of-thought in large multimodal models for reasoning quality, robustness, and efficiency. *Preprint*, arXiv:2502.09621.

Gangwei Jiang, Yahui Liu, Zhaoyi Li, Qi Wang, Fuzheng Zhang, Linqi Song, Ying Wei, and Defu Lian. 2025b. What makes a good reasoning chain? uncovering structural patterns in long chain-of-thought reasoning. *Preprint*, arXiv:2505.22148.

Priyanka Kargupta, Shuyue Stella Li, Haocheng Wang, Jinu Lee, Shan Chen, Orevaoghene Ahia, Dean Light, Thomas L. Griffiths, Max Kleiman-Weiner, Jiawei Han, Asli Celikyilmaz, and Yulia Tsvetkov. 2025. Cognitive foundations for reasoning and their manifestation in llms. *Preprint*, arXiv:2511.16660.

Joongwon Kim, Anirudh Goyal, Liang Tan, Hannaneh Hajishirzi, Srinivasan Iyer, and Tianlu Wang. 2025. Astro: Teaching language models to reason by reflecting and backtracking in-context. *Preprint*, arXiv:2507.00417.

David R. Krathwohl. 2002. A revision of bloom's taxonomy: An overview. *Theory into Practice*, 41(4):212–218.

Abhinav Kumar, Jaechul Roh, Ali Naseh, Marzena Karpinska, Mohit Iyyer, Amir Houmansadr, and Eugene Bagdasarian. 2025. Overthink: Slowdown attacks on reasoning llms. *Preprint*, arXiv:2502.02542.

Ana Kuzle. 2013. Patterns of metacognitive behavior during mathematics problem-solving in a dynamic geometry environment. *International Electronic Journal of Mathematics Education*, 8(1):20–40.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, and 11 others. 2023. Measuring faithfulness in chain-of-thought reasoning. *Preprint*, arXiv:2307.13702.

Ming Li, Lichang Chen, Jiuhai Chen, Shwai He, Jiuxiang Gu, and Tianyi Zhou. 2024a. Selective reflection-tuning: Student-selected data recycling for LLM instruction-tuning. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 16189–16211, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Ming Li, Yanhong Li, and Tianyi Zhou. 2025a. What happened in LLMs layers when trained for fast vs. slow thinking: A gradient perspective. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32017–32154, Vienna, Austria. Association for Computational Linguistics.

Ming Li, Zhengyuan Yang, Xiyao Wang, Dianqi Li, Kevin Lin, Tianyi Zhou, and Lijuan Wang. 2025b. What makes reasoning models different? follow the reasoning leader for efficient decoding. *arXiv preprint arXiv:2506.06998*.

Ming Li, Nan Zhang, Chenrui Fan, Hong Jiao, Yanbin Fu, Sydney Peters, Qingshu Xu, Robert Lissitz, and Tianyi Zhou. 2025c. Understanding the thinking process of reasoning models: A perspective from schoenfeld's episode theory. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 18278–18299, Suzhou, China. Association for Computational Linguistics.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. 2024b. From quantity to quality: Boosting LLM performance with self-guided data selection for instruction tuning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7595–7628, Mexico City, Mexico. Association for Computational Linguistics.

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

Wei Liu, Ruochen Zhou, Yiyun Deng, Yuzhen Huang, Junteng Liu, Yuntian Deng, Yizhe Zhang, and Junxian He. 2025. Learn to reason efficiently with adaptive length-based reward shaping. *Preprint*, arXiv:2505.15612.

Sara Vera Marjanović, Arkil Patel, Vaibhav Adlakha, Milad Aghajohari, Parishad BehnamGhader, Mehar Bhatia, Aditi Khandelwal, Austin Kraft, Benno Krojer, Xing Han Lù, and 1 others. 2025. Deepseek-r1 thoughtology: Let's think about llm reasoning. *arXiv preprint arXiv:2504.07128*.

John Mason, Leone Burton, and Kaye Stacey. 2010. *Thinking Mathematically*, second edition. Pearson Education, Harlow, England.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *Preprint*, arXiv:2501.19393.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, and et al. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

OpenAI. 2024a. Learning to reason with llms.

OpenAI. 2024b. OpenAI o1-mini System Card.

OpenAI. 2024c. OpenAI o1 System Card.

OpenAI. 2025a. Introducing GPT-4.1 in the API.

OpenAI. 2025b. OpenAI GPT-5 System Card.

OpenAI. 2025c. OpenAI o3-mini System Card.

George Pólya. 1945. *How to Solve It*. Princeton University Press, Princeton.

Qwen Team. 2025a. Qwen3: Think deeper, act faster.

Qwen Team. 2025b. Qwq-32b: Embracing the power of reinforcement learning.

Alan H. Schoenfeld. 1985. *Mathematical Problem Solving*. Academic Press.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *Preprint*, arXiv:2408.03314.

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

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, and et al. 2024. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Constantin Venhoff, Iván Arcuschin, Philip Torr, Arthur Conmy, and Neel Nanda. 2025. Understanding reasoning in thinking language models via steering vectors. *Preprint*, arXiv:2506.18167.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, and 1 others. 2025. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Xuyang Wu, Jinming Nian, Ting-Ruen Wei, Zhiqiang Tao, Hsin-Tai Wu, and Yi Fang. 2025. Does reasoning introduce bias? a study of social bias evaluation and mitigation in llm reasoning. *Preprint*, arXiv:2502.15361.

Violet Xiang, Chase Blagden, Rafael Rafailov, Nathan Lile, Sang Truong, Chelsea Finn, and Nick Haber. 2025. Just enough thinking: Efficient reasoning with adaptive length penalties reinforcement learning. *Preprint*, arXiv:2506.05256.

Wei Xiong, Hanning Zhang, Chenlu Ye, Lichang Chen, Nan Jiang, and Tong Zhang. 2025. Self-rewarding correction for mathematical reasoning. *Preprint*, arXiv:2502.19613.

Xiao-Wen Yang, Xuan-Yi Zhu, Wen-Da Wei, Ding-Chu Zhang, Jie-Jing Shao, Zhi Zhou, Lan-Zhe Guo, and Yu-Feng Li. 2025. Step back to leap forward: Self-backtracking for boosting reasoning of language models. *Preprint*, arXiv:2502.04404.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. 2025. Limo: Less is more for reasoning. *Preprint*, arXiv:2502.03387.

Joseph B. W. Yeo and Ban Har Yeap. 2010. Characterising the cognitive processes in mathematical investigation. *International Journal for Mathematics Teaching and Learning*, pages 1–10.

Table of Contents for Appendix

A Related Work

A.1 Cognitive Theories of Math Problem Solving

Analyzing human problem-solving has evolved from broad cognitive taxonomies to domain-specific frameworks. Early models like Bloom's Taxonomy (Krathwohl, 2002) categorized cognitive processes hierarchically but failed to capture the iterative nature of mathematical problem-solving. Similarly, instructional frameworks such as Pólya (1945) four-phase model and subsequent refinements by Mason et al. (2010) and Yeo and Yeap (2010) provided valuable pedagogical scaffolding but lacked the fine-grained operationalization required for rigorous empirical annotation. Even more detailed models like Greenes (1995), while emphasizing metacognition, remained too sequential to effectively code the nonlinear dynamics often observed in real-world reasoning tasks.

Schoenfeld (1985) episode theory offers a robust, empirically validated scheme for coding behaviors into distinct episodes such as *Reading*, *Analysis*, *Exploration*, and *Verification*. Unlike prescriptive models, Schoenfeld's framework explicitly captures the strategic decisions and metacognitive control—or lack thereof—that determine problem-solving success (Kuzle, 2013). This granular focus on cognitive transitions makes it particularly suitable for analyzing AI-generated reasoning, which often struggles with self-regulation. Consequently, Schoenfeld's model provides the necessary precision to systematically annotate and evaluate the complex, often non-linear reasoning traces investigated in this study. Li et al. (2025c) first applied this framework to analyze the reasoning process of large language models. However, they haven't conducted a systematic fine-grained analysis of the episode-level patterns of annotated reasoning traces or compared the episode-level patterns between different models.

A.2 Large Reasoning Models

The surge in LLM development has catalyzed substantial efforts to bolster their reasoning skills (Ahn et al., 2024; Besta et al., 2025; Chen et al., 2025a). Most research has centered on enhancing these abilities via post-training methods. For instance, reinforcement learning has been utilized to steer models towards superior reasoning strategies (Shao et al., 2024; Xiong et al., 2025; Cui et al., 2025). Furthermore, instruction tuning using rigorously selected, high-quality datasets has proven effective in boosting performance (Li et al., 2024b,a; Ye et al., 2025; Muennighoff et al., 2025; Li et al., 2025a). Moreover, Snell et al. (2024); OpenAI (2024a) have shown that scaling up test time compute can also improve the reasoning capabilities of models. Yet, while these models demonstrate remarkable gains in mathematical reasoning on benchmarks, there remains a notable gap in systematically understanding and quantifying how these enhancements alter model behavior. Recent efforts have begun to bridge this gap from an information-theoretic perspective; for example, Wang et al. (2025) emphasize the critical role of high-entropy minority tokens in reasoning traces during reinforcement learning and Chain-of-Thought (CoT) paths. Distinct from their token-level, entropy-based methodology, our work analyzes model output behavior at the semantic and cognitive levels. Despite these divergent starting points, our findings are strikingly aligned: while their analysis reveals the importance of logical connectors within and across sentences, our study similarly highlights the positive contribution of "Monitor"-related phase transitions in the reasoning process.

A.3 Efficient Reasoning Methods

Overthinking (Chen et al., 2025b; Fan et al., 2025) has been an identified issue of reasoning models. They tend to produce excessive intermediate steps or consider unnecessary details, which can lead to inefficient reasoning. To address this issue, several methods (Sui et al., 2025; Feng et al., 2025a) have been proposed to encourage the reasoning models to reason more efficiently. Specifically, L1 (Aggarwal and Welleck, 2025a) proposes a simple reinforcement learning method that optimizes for accuracy and adherence to user-specified length constraints. ThinkPrune (Hou et al., 2025) offers a simple solution that continuously trains the long-thinking LLMs via reinforcement learning with an added token limit. Arora and Zanette (2025) train reasoning models to dynamically allocate inference-time compute based on task complexity. More works (Fang et al., 2025; Liu et al., 2025; Xiang et al., 2025; Li et al., 2025b) have been proposed to encourage the efficient reasoning of models. However, currently, very limited work explicitly analyzes how these methods are different in behavior or episode-level patterns. Our work is the first to systematically analyze the episode-level patterns of these methods in comparison to

the baseline reasoning models.

A.4 Reasoning Analysis Methods

Chain-of-thoughts (CoT) (Wei et al., 2023) can significantly elicit the reasoning capabilities of models. Various works have studied the different properties of CoT, including bias (Wu et al., 2025), faithfulness (Lanham et al., 2023), and redundancy (Chen et al., 2025b). Specifically, for o1-like reasoning models that have particularly long reasoning traces and showcase more system-II level reasoning abilities, efforts have been made to uncover the structural patterns of CoT (Jiang et al., 2025b). Bogdan et al. (2025) introduced a black-box method that measures each sentence's counterfactual importance to understand the sentence-level importance in long CoT chains. Feng et al. (2025b) introduced a graph view of CoT to extract structure and identified an effective statistic that correlated to the correctness of the reasoning trace. Li et al. (2025c) and Kargupta et al. (2025) incorporated theories from cognitive psychology to label the episodes of CoT in analogy of human problem-solving processes.

Our work, built upon Li et al. (2025c), is the first to systematically analyze the statistical patterns of reasoning traces grounded in cognitive theories. While our work builds on a similar episode taxonomy, it introduces several key extensions. Rather than directly reusing the taxonomy in Li et al. (2025c), we refine it to improve scalability and alignment with standardized LLM reasoning benchmarks by removing the hierarchical structure and introducing an explicit Answer state for practical applicability. In addition, whereas Li et al. (2025c) focuses on SAT-style data, we conduct our experiments on Omni-Math (Gao et al., 2024), a widely used dataset in contemporary LLM reasoning research, making our findings more representative of current evaluation settings. Beyond annotation, we perform a substantially deeper quantitative analysis of episode dynamics, including episode divergence, temporal evolution, coverage, and multi-step transition patterns. We further extend prior work by introducing episode-level n-gram pattern mining with mutual information, whereas Li et al. (2025c) primarily analyzes one-step transitions. Finally, we demonstrate broader downstream applications of the framework by using logistic regression to examine how episode-level features correlate with correctness, and by analyzing efficient reasoning methods to identify which behavioral patterns are selectively suppressed. Taken together, although both works are grounded in the same underlying cognitive theory, our approach represents a refined and extended framework with deeper analysis and stronger practical applicability.

Concurrent with our work, Kargupta et al. (2025) also investigate cognitive behaviors in reasoning traces, with particular attention to success-related elements and temporal patterns. While both studies are grounded in cognitive science, they differ in scope and level of abstraction. Kargupta et al. (2025) propose a taxonomy structured along four dimensions comprising 28 fine-grained cognitive elements, enabling detailed identification of localized reasoning behaviors. In contrast, our framework organizes reasoning into a smaller set of eight episodes that emphasize the overall structure and progression of problem solving. As such, their approach provides higher granularity at the micro level, whereas ours focuses on capturing the global dynamics of reasoning processes. We view these perspectives as complementary, collectively contributing to a more comprehensive understanding of cognitive behavior in large language model reasoning.

B Implementation Details

B.1 Full Model List

We conduct our analysis on a variety of reasoning and non-reasoning models. The reasoning models include DeepSeek-R1 (DeepSeek-AI et al., 2025), DeepSeek-R1-Distill-Qwen, QwQ-32B (Qwen Team, 2025b), Phi4 (Abdin et al., 2024), Qwen3-32B (Qwen Team, 2025a), Gemini-2.5-Flash (Comanici et al., 2025), GPT-o1-mini (OpenAI, 2024b) and GPT-o3-mini (OpenAI, 2025c). The non-reasoning models include GPT-4o (OpenAI et al., 2024), Gemini-2.0-Flash (Team et al., 2024), Qwen2.5-32B (Team, 2024) and the non-reasoning mode of Phi4 and Qwen3-32B. We also study some efficient reasoning models including L1 (Aggarwal and Welleck, 2025b), ThinkPrune (Hou et al., 2025), and models released by Arora and Zanette (2025).

B.2 Temporal Dynamics Details

To investigate the temporal dynamics of reasoning phases across model responses, we analyze how the distribution of cognitive phases evolves over the course of generation. For each annotated response, we divide the sequence into B equal-sized temporal bins based on token position, where $B = 25$ in our analysis. Within each bin, we compute the frequency of tokens belonging to each reasoning phase category. Specifically, for a response with total length L tokens, we define bin size as $\Delta = L/B$. Each token at position t is assigned to bin $b = \lfloor t/\Delta \rfloor$, ensuring uniform temporal coverage across responses of varying lengths. For each category c and bin b, we count the number of tokens belonging to that category, yielding a raw frequency distribution $f_{c,b}$.

To enable comparison across responses with different phase compositions, we normalize the frequency distribution within each response. For category c in response r, we compute the normalized frequency in bin b as:

$$\tilde{f}_{c,b}^{(r)} = \frac{f_{c,b}^{(r)}}{\sum_{b'=1}^{B} f_{c,b'}^{(r)}} \tag{3}$$

where the denominator represents the total number of tokens of category c in response r. This normalization ensures that the distribution sums to 1 for each category within each response, allowing us to examine the relative temporal positioning of phases independent of their absolute frequency.

B.3 Transition Patterns Details

After annotation, each response can be represented with a sequence of cognitive phases. To study which phase combinations are specific to a certain model or kinds of models (e.g. Deepseek vs others, reasoning vs non-reasoning), we conduct the phase-based N-gram analysis that **treats each phase as a gram** and investigate what combinations of grams are most significant patterns of a type of models. Specifically, we use a letter to represent each phase (e.g. R for READ), and each response becomes a string. We use the mutual information to identify the most discriminative patterns between two groups:

$$I(X;G) = \sum_{x\in\{0,1\}} \sum_{g\in\mathcal{G}} p(x,g) \log \frac{p(x,g)}{p(x)\,p(g)}, \tag{4}$$

where $x \in \{0, 1\}$ indicates whether a given episode N-gram is present in a trace, and g denotes the source group. Since MI is symmetric and only measures the strength of association, we further use conditional probability to determine which group each top-k discriminative pattern is attributed to.

C Confusion Matrix

To further examine annotation quality, we report the confusion matrices of the four candidate annotation models on the same set of 7,067 human-annotated sentences in Tables 8–11. Rows correspond to gold labels and columns correspond to model predictions. Across all models, the matrices are strongly diagonally dominant, indicating that most sentences are assigned to the correct episode category.

Several common patterns appear across all four annotators. First, the dominant confusions consistently involve behaviorally adjacent categories, especially *Analyze* → *Implement*, *Verify* → *Implement*, and, to a lesser extent, spillover from *Verify* into other active reasoning categories such as *Analyze*, *Monitor*, and *Plan*. Second, *Answer* and *Read* remain comparatively easy to identify, with much cleaner diagonals than the more process-oriented categories. Third, the stronger annotators mainly differ from the weaker ones not in the types of errors they make, but in how often they make them: GPT-5 and GPT-4.1 preserve the same overall structure while showing less off-diagonal mass than the two Gemini models. For GPT-5 specifically, the largest off-diagonal entries are *Analyze* → *Implement* (225 cases) and *Verify* → *Implement* (89 cases), which account for only 3.2% and 1.3% of the full dataset, respectively, or 4.4% combined. Therefore, we believe the resulting annotations are sufficiently reliable for the downstream analyses in the main paper, including the temporal "heartbeat" patterns.

	Analyze	Answer	Explore	Implement	Monitor	Plan	Read	Verify
Analyze	1031	14	35	225	7	10	18	67
Answer	1	405	2	4	2	1	0	1
Explore	6	2	594	13	10	14	0	15
Implement	15	15	3	2542	0	4	1	86
Monitor	17	1	16	10	322	17	9	53
Plan	14	3	18	79	42	584	4	30
Read	6	0	2	14	1	6	259	1
Verify	25	18	7	89	8	1	4	996

Table 8: Confusion matrix of the GPT-5 annotation model on the 7,067 human-annotated sentences. Rows denote ground-truth labels and columns denote predicted labels.

	Analyze	Answer	Explore	Implement	Monitor	Plan	Read	Verify
Analyze	1219	3	12	101	5	30	24	13
Answer	15	389	1	5	0	2	0	4
Explore	46	1	514	11	26	48	5	3
Implement	102	9	5	2448	2	46	9	45
Monitor	5	1	6	2	409	14	2	6
Plan	18	0	7	42	13	680	12	2
Read	5	0	0	5	2	3	274	0
Verify	151	8	17	56	64	67	3	782

Table 9: Confusion matrix of the GPT-4.1 annotation model on the 7,067 human-annotated sentences.

	Analyze	Answer	Explore	Implement	Monitor	Plan	Read	Verify
Analyze	1109	4	34	149	17	20	11	63
Answer	7	374	4	20	0	7	0	4
Explore	30	0	565	6	14	19	0	20
Implement	68	1	8	2389	2	15	0	183
Monitor	17	3	31	9	299	22	2	62
Plan	22	4	36	62	13	593	5	39
Read	36	0	2	17	6	9	214	5
Verify	95	11	17	57	33	13	0	922

Table 10: Confusion matrix of the Gemini-2.5-Flash annotation model on the 7,067 human-annotated sentences.

	Analyze	Answer	Explore	Implement	Monitor	Plan	Read	Verify
Analyze	1140	0	13	146	0	13	3	92
Answer	15	345	2	29	1	7	0	17
Explore	72	1	455	16	7	43	0	60
Implement	94	0	3	2423	0	18	1	127
Monitor	22	2	13	14	229	31	1	133
Plan	69	1	10	55	7	580	5	47
Read	47	0	1	10	2	5	214	10
Verify	92	2	3	117	2	9	0	923

Table 11: Confusion matrix of the Gemini-2.5-Pro annotation model on the 7,067 human-annotated sentences.

D Question-Domain Analysis

To assess whether the "heartbeat" pattern in Figure 3 is driven only by aggregate averaging or also persists across different kinds of math problems, we analyze episode-level allocation separately for each major question domain in Omni-Math. Table 12 reports the average proportion of tokens assigned to each episode category for each domain.

Several common patterns emerge across domains. First, *Analyze* and *Implement* consistently take the largest shares, indicating that most domains devote the bulk of reasoning to problem interpretation and execution. Second, *Read* and *Answer* remain comparatively small in every domain, suggesting that the main differences do not come from input parsing or final response formatting. Third, the largest domain-specific variation appears in the more execution- and control-oriented phases: *Implement* is higher in Calculus and Number Theory, *Verify* is especially prominent in Geometry and Precalculus, and *Monitor* is notably higher in Applied Mathematics. Overall, these results suggest that the main "heartbeat" pattern is not an artifact of averaging across unrelated problem types; rather, it reflects a broadly shared reasoning structure with moderate domain-specific adjustments.

Question Domain	Read	Analyze	Plan	Implement	Explore	Verify	Answer	Monitor
Algebra	3.04	22.97	8.68	32.29	11.77	13.00	3.17	5.09
Calculus	3.37	19.38	9.60	35.57	10.57	12.90	3.51	5.09
Applied Mathematics	3.41	22.17	7.11	28.49	12.10	11.42	2.27	13.02
Discrete Mathematics	3.26	23.99	7.46	28.32	11.67	14.01	2.79	8.49
Geometry	3.18	21.56	8.23	30.42	9.69	19.58	2.26	5.09
Number Theory	2.94	22.00	8.29	35.31	10.35	13.47	2.76	4.87
Precalculus	2.90	20.62	9.51	27.99	11.82	16.30	5.56	5.32

Table 12: Average episode-level token allocation (%) across question domains in Omni-Math. The domain-wise distributions are broadly consistent, while still revealing moderate variation in execution- and checking-related phases.

E Episode-level Diagnostics of Correctness Details

In this section, we provide the detailed implementation of the correctness diagnostic analysis presented in Section 4.1. We formulate the problem as a binary classification task, where we predict the correctness of a reasoning trace based on its episode-level cognitive features.

E.1 Feature Engineering.

For each reasoning trace, we extract a comprehensive set of features capturing global statistics, episode intensity, and transition dynamics. Let $S = \{s_1, s_2, \ldots, s_N\}$ be the sequence of sentences in a trace, where each sentence s_i is associated with an episode tag $e_i \in \mathcal{E}$ and a token count t_i. The set of episode categories $\mathcal{E}$ consists of the 8 refined episodes: *Read, Analyze, Plan, Implement, Explore, Verify, Monitor, Answer*. We compute the following feature groups:

- **Global Statistics**:
  - *Total Tokens*: The total number of tokens in the response, $\sum t_i$, estimated using the GPT-4 tokenizer ('cl100k_base').
  - *Think Ratio*: The proportion of tokens belonging to the reasoning process (excluding the final *Answer* content) relative to the total tokens.
- **Episode Intensity (Token Ratios)**: For each episode category $c \in \mathcal{E}$, we compute the proportion of the total budget allocated to it:

$$\text{Ratio}_c = \frac{\sum_{i:e_i=c} t_i}{\sum_j t_j}$$

  This yields 8 features representing the relative dominance of each cognitive behavior.
- **Transition Features**: We construct a transition matrix capturing the frequency of shifts between reasoning states. We compute the raw count of transitions from episode src to episode tgt:

$$\text{Trans}_{src \to tgt} = \sum_{i=1}^{N-1} \mathbb{I}(e_i = src \wedge e_{i+1} = tgt)$$

  This results in $8 \times 8 = 64$ transition features, capturing the structural flow of reasoning (e.g., *Explore* $\rightarrow$ *Verify*).

E.2 Model Specification.

We employ a Lasso-regularized Logistic Regression model to identify the most predictive features while enforcing sparsity for interpretability. The model predicts the probability of correctness $p(y = 1|\mathbf{x})$:

$$p(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b)$$

where $\mathbf{x}$ is the standardized feature vector, $\mathbf{w}$ are the learned coefficients, and $\sigma(\cdot)$ is the sigmoid function. To handle the high-dimensional feature space (particularly the transition matrix) and select only the most robust signals, we use L1 regularization (Lasso). The objective function is:

$$\min_{\mathbf{w},b} \Bigg(-\frac{1}{M} \sum_{j=1}^{M} \Big[y_j \log \hat{y}_j + (1 - y_j) \log(1 - \hat{y}_j) \Big] + \lambda \|\mathbf{w}\|_1 \Bigg)$$

where M is the number of samples and λ controls the regularization strength.

Implementation Details. We use the reasoning traces from 5 representative open-source reasoning models (DeepSeek-R1, DeepSeek-R1-Distill-Qwen-32B, Phi-4, Qwen3-32B, and QwQ-32B). All features are standardized using Z-score normalization before training. This ensures that the magnitude of coefficients directly reflects the relative importance of features. We use the `liblinear` solver which is well-suited for smaller datasets with L1 regularization. We set the regularization parameter $C = 0.5$ (where $C = 1/\lambda$) to promote feature selection, and use a maximum of 2,000 iterations to ensure convergence. We analyze the learned coefficients $\mathbf{w}$. Features with positive coefficients increase the probability of a correct answer, while those with negative coefficients are associated with incorrect outcomes. Features with zero coefficients are pruned by the Lasso regularization, indicating they are less relevant for predicting correctness in this linear approximation.

E.3 Full Feature Coefficients.

Table 13 shows the full feature coefficients for the Lasso logistic regression model in the diagnostic Lasso logistic regression case study

Positive Contributors			Negative Contributors		
#	**Feature**	β	**#**	**Feature**	β
1	Trans (Exp → Mon)	+0.41	1	Ratio (Exp)	-0.54
2	Trans (Exp → Ana)	+0.31	2	Trans (Exp → Ver)	-0.45
3	Trans (Mon → Ana)	+0.28	3	Trans (Exp → Ans)	-0.41
4	Trans (Read → Ver)	+0.27	4	Count (Total)	-0.36
5	Ratio (Think)	+0.22	5	Trans (Imp → Read)	-0.33
6	Trans (Ans → Ver)	+0.22	6	Trans (Imp → Imp)	-0.28
7	Ratio (Ver)	+0.21	7	Trans (Ana → Mon)	-0.26
8	Trans (Read → Mon)	+0.18	8	Trans (Ver → Read)	-0.25
9	Trans (Read → Read)	+0.17	9	Trans (Ver → Plan)	-0.25
10	Ratio (Mon)	+0.16	10	Trans (Mon → Read)	-0.22
11	Trans (Read → Imp)	+0.15	11	Ratio (Ans)	-0.21
12	Trans (Exp → Read)	+0.14	12	Trans (Plan → Mon)	-0.19
13	Trans (Plan → Ver)	+0.12	13	Trans (Read → Ans)	-0.18
14	Trans (Plan → Plan)	+0.12	14	Trans (Imp → Ans)	-0.15
15	Ratio (Plan)	+0.10	15	Trans (Ana → Ana)	-0.14
16	Trans (Imp → Mon)	+0.07	16	Ratio (Read)	-0.13
17	Trans (Ana → Ver)	+0.07	17	Trans (Ans → Ana)	-0.11
18	Trans (Ver → Imp)	+0.05	18	Trans (Read → Exp)	-0.11
19	Trans (Mon → Ver)	+0.05	19	Trans (Ver → Mon)	-0.08
20	Trans (Ana → Plan)	+0.04	20	Trans (Exp → Plan)	-0.08
21	Trans (Mon → Mon)	+0.04	21	Trans (Plan → Imp)	-0.06
22	Trans (Plan → Read)	+0.03	22	Trans (Ans → Imp)	-0.05
23	Trans (Plan → Exp)	+0.01	23	Trans (Ans → Exp)	-0.05
			24	Trans (Imp → Ana)	-0.03
			25	Trans (Read → Plan)	-0.03
			26	Trans (Ans → Mon)	-0.02
			27	Trans (Ver → Ana)	-0.01

Table 13: The full predictive episode-level features for correctness in the diagnostic Lasso logistic regression case study. Features are ranked by their coefficient (β) magnitude.

F Detailed ThinkARM Framework

The detailed ThinkARM Framework is shown in Figure 4. For each batch of sentences in the model response, the annotation model tags the episode category for each sentence based on the guidebook, question, previous context and outputs the rationale and annotation in JSON format as instructed by the format prompt. The labels after batch processing are then concatenated to form the labels for the whole response. The guidebook and prompt template in the figure are in Appendix G and Appendix H.

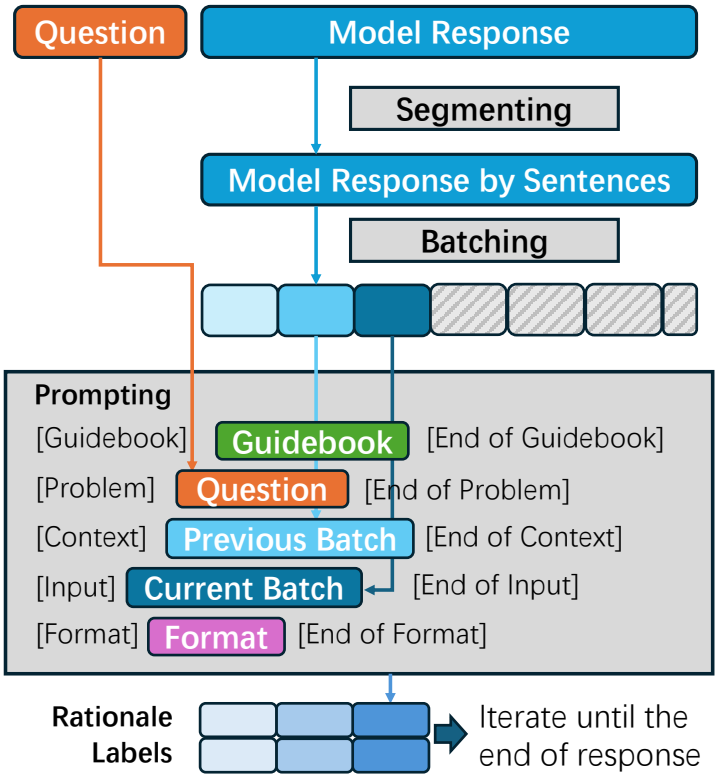

Figure 4: The ThinkARM Framework. For each question-response pair, the model response is first segmented into sentences. They are then tagged by the annotation models in batches, along with information about the guidebook, question, context, and format. The guidebook is in Appendix G, the prompt template is in Appendix H.

G Refined Annotation Guidebook

In this project, we aim to analyze the reasoning process of current large language models (LLMs) with advanced reasoning capabilities, i.e., Large Reasoning Models (LRMs), based on a modified version of Alan Schoenfeld's (1985) "Episode-Timeline" framework for problem-solving. The original Schoenfeld theory was built on hundreds of hours of recorded tapes of students tackling non-routine math problems while being asked to think aloud. Widely regarded as a gold-standard framework in mathematics education research, this theory offers a rigorously validated, fine-grained lens for dissecting both expert and novice problem-solving strategies. After thorough investigation, we find that the thinking process of LRMs can be well-aligned with the episodes in the theory, as they also follow similar problem-solving processes. Thus, in this project, we aim to annotate the model (solver) responses with these episode categories. To better apply the theory to the analysis of model responses, we utilize sentence-level annotation, which is used to capture the fine-grained behavior of each sentence, including **eight** categories: Read, Analyze, Plan, Implement, Explore, Verify, Monitor, and Answer. The original Schoenfeld theory only has six categories: Read, Analyze, Plan, Implement, Explore, and Verify. These categories describe the thinking behaviors of how humans solve a problem. Later, an additional "Monitor" category was included in the system to capture behaviors that do not contain specific content but are still important, such as "Let me think." Moreover, when trying to apply the theory to analyzing LLM behaviors, we introduce another category, "Answer," to represent the sentence that delivers the answer. Thus, in total, there are eight categories. For each sentence, the annotation depends on both the current sentence itself and its context.

G.1 Label Definitions and Guidelines

1. Read

- **Definition:** This is usually the initial phase, which focuses on extracting or restating the given information, conditions, and the goal of the problem as presented. It involves understanding the question without any inference of strategy or reasoning.
- **Guidelines:**
  - Sentences in this category should directly present the content of the original problem statement.
  - Look for phrases that recall or repeat elements of the question.
  - This label is mostly presented for the model's initial processing of the problem.
- **Potential Keywords/Indicators:** "The question asks...", "The problem requires...", "We are given...", "The goal is to...", "The choices are...", direct quotes from the problem.
- **Distinguishing Features:**
  - This stage is purely about understanding the input, not about processing it or deciding how to solve it. It should not contain content like trying to understand or analyze the question.
  - Avoid labeling sentences as Read if they include any form of analysis or evaluation of the problem. The Read stage usually appears at the beginning of the reasoning. However, it can also appear in the middle of the reasoning, in order to ensure that the question was understood correctly.
- **Example:** "The question asks us to find the value of x in the equation $2x + 5 = 10$."

2. Analyze

- **Definition:** This stage involves constructing or recalling relevant theories, introducing necessary symbols, and deducing relationships based on the problem statement and existing knowledge. The core activity is explanation or logical inference that sets the stage for the solution but does not involve concrete calculations yet.
- **Guidelines:**
  - Sentences should explain the underlying mathematical concepts or principles relevant to the problem.
  - This category includes analysis of either the problem itself or intermediate results.
  - This label applies to logical deductions and inferences made with certainty.

- **Potential Keywords/Indicators:** "According to...", "We can define...", "This implies that...", "Therefore...", "Based on this...", "We can infer that...", "Let's note that...", "Let me observe that...", "Let's recall that..."
- **Distinguishing Features:**
  - The Analyze episode involves certain inferences and explanations, unlike Explore, which shows uncertainty.
  - The actual execution of calculations is mainly in the Implement stage.
  - Analyze does not involve any concrete calculation, which is unlike Implement.
  - The behavior of setting some basic notations should be in the Implement stage, e.g., "Let me set $a = 1$, then...".
- **Important Note:** Be careful not to include sentences that involve substituting values or performing calculations, as those belong to the Implement stage.
- **Example:** "According to the Pythagorean theorem, in a right-angled triangle, the square of the hypotenuse is equal to the sum of the squares of the other two sides." or "If I can get the equation in slope-intercept form ($y = mx + b$), then I can plug in $y = 4$ and solve for x, which should be d."

3. Plan

- **Definition:** This stage involves announcing the next step or outlining the entire solution strategy. It represents a commitment to a particular course of action before the actual execution begins.
- **Guidelines:**
  - Sentences should clearly state the intended next step or the overall plan.
  - Look for explicit declarations of intent, often using the first person or imperative voice.
  - This stage signifies that a decision has been made on how to proceed, and the next step should be related to math problem solving, rather than generally saying "let's think about it."
- **Potential Keywords/Indicators:** "Next, we will...", "The next step is to...", "We need to...", "Let's proceed by...", "I will now...", "The plan is to...", "We should first...", "To..., do...", "The xxx we need/want is...", "Let's...", "Then/Now calculate/consider...".
- **Distinguishing Features:**
  - The Plan phase clearly indicates the intended action, unlike Analyze, which explains concepts, or Explore, which suggests possibilities.
  - It precedes the actual carrying out of the plan in the Implement stage.
  - Note that sentences like "Let's denote..." are Analyze, because this is introducing a new variable, rather than making a plan.
  - Sentences like "let's verify...", "let's test..." or "let's double-check" are Verify.
- **Example:** "Next, we will differentiate both sides of the equation with respect to x."

4. Implement

- **Definition:** This stage is the operational phase where the planned strategy is executed. It involves setting up basic notations, performing specific calculations, constructing diagrams, enumerating possibilities, or coding solutions using numerical values, symbols, or geometric objects.
- **Guidelines:**
  - Sentences should describe the actual steps taken to solve the problem.
  - Look for mathematical operations, substitutions, and the generation of intermediate results.
  - This stage is about "doing" the math.
- **Potential Keywords/Indicators:** "Substituting $x = 2$, we get...", "Therefore, $P(1) = -1$", "Expanding the expression...", "The matrix becomes...", "Let me set $a = 1$, then...", "Let's denote $x = 10$...", actual mathematical equations and calculations.
- **Distinguishing Features:**
  - Implement involves concrete actions and calculations, unlike Analyze, which focuses on theoretical explanations, or Plan, which outlines future actions.

- If a conclusion follows the implementation of math, that conclusion is tagged as Implement, such as "therefore, the sum of all possible values is 5."

- **Example:** "Substituting $x = 3$ into the equation, we get $2(3) + 5 = 6 + 5 = 11$."

5. Explore

- **Definition:** This stage is characterized by generating potential ideas, making guesses, drawing analogies, or attempting trial calculations that might be abandoned later. The model is exploring different avenues without committing to a specific solution path. This stage often involves uncertainty.

- **Guidelines:**
  - Sentences should suggest alternative approaches or possibilities.
  - Look for tentative language and expressions of uncertainty.
  - This stage involves brainstorming and initial investigations without a clear commitment to a particular method.

- **Potential Keywords/Indicators:** "Maybe we can try...", "Perhaps we could use...", "What if we consider...", "Another possibility is...", "Could this be related to...", "Maybe I should...", "Maybe there is another way...", "Maybe we can try...", "Maybe there is a better way...", "Maybe consider...", "Perhaps... is...", "Let's try...", "Alternatively, maybe use...", "Wait, but maybe...", "But in soccer, it's possible to lose a game but still have more total goals?"; question marks indicating uncertainty about a step.

- **Distinguishing Features:**
  - Explore is marked by uncertainty and a lack of commitment, unlike Plan, which announces a definite course of action.
  - It involves considering various options before settling on a specific plan. If a sentence contains analyzing the problem, implementing the calculation, or verifying the result or thought, even if it follows sentences like "Maybe we can try...", the sentences are not considered Explore at the sentence level, and therefore should not be labeled as Explore. Rather, these sentences are considered Analyze, Implement, or Verify within the Explore episode at the paragraph level. Only sentences like "Maybe we can try..." will be labeled as Explore at the sentence level.

- **Example:** "Maybe we can try substituting different values for x to see if we can find a pattern."

6. Verify

- **Definition:** This stage involves judging the correctness, effectiveness, or simplicity of the obtained result or the method used. It might include checking the answer, using an alternative method for calculation, or estimating bounds.

- **Guidelines:**
  - Sentences should express an evaluation or confirmation of the solution or the process.
  - Look for keywords related to checking, confirming, or validating.
  - This stage ensures the solution and result are accurate and make sense.

- **Potential Keywords/Indicators:** "Let me double-check...", "This is consistent with...", "Plugging it back in...", "Therefore, the answer is correct.", "Let's confirm...", "Let me check again...", "We can confirm this by...", "This result seems reasonable because...", "The answer is...?", "Is the answer...?", "Is there any mistake?", "Did I make a mistake?", "This is the same/correlated as previous...", "But this seems to contradict...", "...lead/arrive to the same answer", "Wait, we don't know... yet", "Let's try another way to verify...", "XXX is possible/impossible." When the following sentences are meant as conclusions, "...is indeed...", "...should be..."

- **Distinguishing Features:**
  - Verify focuses on evaluating the solution, unlike Implement, which focuses on generating it.
  - It often involves comparing the result with initial conditions or using alternative methods.

- **Example:** "Let me double-check my calculations: $2 \times 3 + 5 = 11$, which matches the previous result."

7. Monitor

- **Definition:** This additional category captures sentences that are typically short interjections or expressions indicating the model's self-monitoring, hesitation, or reflection at the juncture between different episodes. These often do not contain substantial problem-solving content and are brief pauses in the thought process.
- **Guidelines:**
  - Sentences should be short phrases indicating a shift in thought or a brief pause.
  - Look for expressions of uncertainty, reflection, or transition.
  - This label is for meta-comments that don't fit neatly into the other problem-solving stages.
- **Potential Keywords/Indicators:** "Hmm...", "Wait...", "Let me think.", "Okay...", "Let's see.", "Hold on.", "But wait, hold on."
- **Distinguishing Features:**
  - Monitor sentences lack the substantive content of the other categories and primarily serve as indicators of the model's internal processing flow.
  - They are often very short and act as bridges between more content-heavy stages.
  - In most cases, it should not contain evaluation for previous steps or contain any specific question or solution content. If it contains specific content, e.g., "Wait, the problem says...", it should be categorized into others like Read.
- **Example:** "Wait."

8. Answer

- **Definition:** This stage is used for sentences that explicitly state an answer or conclusion to the problem. These sentences deliver the result, either as a final answer at the end of the response or as an intermediate answer that may be subject to later verification or revision. Note: it should be the answer to the given problem, rather than an intermediate answer for a calculation step.
- **Guidelines:**
  - Sentences should directly present a solution, value, or conclusion in response to the given problem statement.
  - Look for clear, declarative statements that summarize the outcome of the reasoning or calculation.
  - This category applies whether the answer is final or provisional.
- **Potential Keywords/Indicators:** "The answer is...", "Hence, the result is...", "So, the final answer is...".
- **Distinguishing Features:**
  - Answer sentences are characterized by their directness in providing a result to the given problem, unlike Verify, which focuses on checking correctness, or Implement, which details the process of obtaining the result.
  - These sentences often appear at the end of a solution but can also occur mid-response as provisional answers.
- **Example:** "Therefore, the answer is 24."

G.2 Important Considerations for Annotators

- **Sentence-Level Focus:** Annotate each sentence individually based on its primary function within the problem-solving process.
- **Context is Key:** While keywords can be helpful, always consider the context of the sentence within the overall response. A sentence might contain a keyword but function differently based on the surrounding text.
- **Refer to Examples:** The examples provided in this guidebook and any additional examples you encounter should serve as valuable references.

H Annotation Prompt

We use the prompt template in Figure 5 to annotate the reasoning traces in our study. The detailed content of the guidebook can be found in Appendix G.

Prompt Template for Annotation

In this project, we aim to analyze the reasoning process of current large language models (LLMs) with advanced reasoning capabilities, i.e., Large Reasoning Models, LRMs, based on a modified version of Alan Schoenfeld's (1985) "Episode-Timeline" framework for problem-solving. Given the model response you need to annotate the sentence-level behavior of the model response with the eight categories: Read, Analyze, Explore, Plan, Implement, Verify, Monitor, and Answer.

The [Guidebook] - [End of the Guidebook] section provides the detailed introduction and definition of each category. The [Math Problem] - [End of the Math Problem] section provides a math problem.
The [Overall Response] - [End of the Overall Response] section provides the overall response of the model to the math problem.
The [Previous Context] - [End of the Previous Context] section provides all the previous context of the response that has been annotated and their corresponding labels.
The [Input] - [End of the Input] section provides the sentences that need to be annotated.
The [Format] - [End of the Format] section provides the format of the output.

[Guidebook]
(Guidebook Content)
[End of Guidebook]
[Math Problem]
(The Question)
[End of the Math Problem]
[Previous Context]
The previous sentences are:**(Previous batch sentences)**
[End of Previous Context]
[Input]
The following sentences that you need to classify:**([1] xxx [2] xxx ... [batch_num] xxx)**
[End of Input]
[Format]
You should format the output in JSON format regarding the index, a short rationale and the fine-grained class of the indexed sentence. The format is as follows:"
{'sentences': [
{'index' 'The index of the sentence', 'reason': 'The short reason of the classification', 'category': 'The fine-grained class of the sentence'},
{'index': 'The index of the sentence', 'reason': 'The short reason of the classification', 'category': 'The fine-grained class of the sentence'},
...
]}"
[End of Format]

Now, annotate the sentences in the [Input] - [End of the Input] section. Refer to the guidebook to make the decision. Strictly follow the index number of the sentence in the [Input] - [End of the Input] section for labeling. You should output the label for batch_num sentences.

Figure 5: The prompt template we used to annotate the reasoning episode.