ABSTRACT

Title of Thesis: INVESTIGATING REASONING MODELS: OVERTHINKING AND THINKING EPISODE THEORY

Chenrui Fan
Master of Science, 2026

Thesis Directed by: Professor Soheil Feizi
Department of Computer Science

Large language models (LLMs) have made remarkable progress in complex reasoning tasks by generating explicit chains of thought. However, the quality, efficiency, and structure of these reasoning processes remain poorly understood. This thesis investigates reasoning models through three complementary studies.

First, we identify a critical failure mode called Missing Premise Overthinking (MiP-Overthinking), where reasoning models generate excessively long responses to ill-posed questions lacking necessary premises. We show that this phenomenon contradicts the test-time scaling law: more computation does not help models identify the missing information. Surprisingly, non-reasoning models demonstrate better critical thinking ability in these scenarios.

Second, we introduce a novel analytical framework grounded in Schoenfeld's Episode Theory, a cognitive science framework originally developed for understanding human mathematical problem-solving. By annotating reasoning traces with cognitive episode labels such as Read, Analyze, Plan,

Implement, Explore, Verify, and Monitor, we provide the first principled methodology for dissecting the reasoning process of large reasoning models.

Third, we scale this episode-level analysis into ThinkARM (Anatomy of Reasoning in Models), a framework that enables automatic, large-scale annotation and comparison of reasoning traces across 15 diverse models. Our analysis reveals consistent temporal organization in reasoning traces, identifies structural differences between reasoning and non-reasoning models, and demonstrates that episode-level patterns correlate with solution correctness and the behavioral impact of efficient reasoning methods.

Together, these three studies advance our understanding of how reasoning models think, where they fail, and how their reasoning processes can be systematically characterized.

INVESTIGATING REASONING MODELS: OVERTHINKING AND THINKING EPISODE THEORY

by

Chenrui Fan

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2026

Advisory Committee:
Professor Soheil Feizi, Chair
Professor Sarah Wiegreffe
Professor Tianyi Zhou

Dedication

To my family and friends.

Acknowledgments

I am deeply grateful to my committee members, Professor Soheil Feizi, Professor Sarah Wiegreffe, and Professor Tianyi Zhou, for their guidance and encouragement in both research and daily life. Their patience and insight have shaped how I approach problems, and I feel fortunate to have learned from them. I look forward to continuing to learn from them as I begin my doctoral studies.

I also owe a great deal to my collaborators, whose ideas and effort shaped the work in this thesis. I am especially thankful to Ming Li and Yize Cheng for their central role in the projects I describe here. I am equally grateful to the friends and colleagues who shared ideas and conversations with me during my master's program, Ziyue Li, Yijun Liang, Xirui Li, and Dongping Chen, and to Professor Pan Zhou at HUST and Professor Lichao Sun at Lehigh University, who first opened the door to research for me.

Finally, I thank my family for their steady love and support, which sustained me through this degree and made the work meaningful.

Table of Contents

[1] This chapter is based on a paper co-authored with Ming Li, Lichao Sun, and Tianyi Zhou, with equal contribution from Ming Li and me [23].

[2] This chapter is based on a paper co-authored with Ming Li, Nan Zhang, Hong Jiao, Yanbin Fu, Sydney Peters, Qingshu Xu, Robert Lissitz, and Tianyi Zhou, with equal contribution from Ming Li, Nan Zhang, and me [56].

[3]This chapter is based on a paper co-authored with Ming Li, Yize Cheng, Soheil Feizi, and Tianyi Zhou, with equal contribution from Ming Li, Yize Cheng, and me [52].

List of Tables

List of Figures

List of Abbreviations

CoT	Chain of Thought
DPO	Direct Preference Optimization
GRPO	Group Relative Policy Optimization
LLM	Large Language Model
MiP	Missing Premise
MMLU	Massive Multitask Language Understanding
RL	Reinforcement Learning
SFT	Supervised Fine-Tuning
ThinkARM	Thinking Anatomy of Reasoning in Models

Chapter 1: Introduction

Large language models (LLMs) have emerged as powerful tools for complex reasoning tasks [20, 35, 72]. By generating explicit chains of thought (CoT) [104], models such as OpenAI's GPT-o1 and the open-source DeepSeek-R1 have demonstrated remarkable performance improvements on mathematical reasoning, code generation, and scientific problem-solving. These reasoning models are typically trained through reinforcement learning or supervised fine-tuning to produce detailed, multi-step reasoning paths, whose increased length is generally associated with improved performance — a phenomenon known as the test-time scaling law [92].

Despite these advances, the reasoning processes of these models remain poorly understood. We know that they can produce long reasoning traces, but several fundamental questions remain open: Does more computation always help? What happens when models face ill-posed problems? Are there meaningful patterns within these reasoning traces? What distinguishes the thinking process of a reasoning model from a non-reasoning one? This thesis addresses these questions through three complementary studies, each forming a chapter.

Chapter 2: MiP-Overthinking. We begin by identifying a critical failure mode of reasoning models. When presented with ill-posed questions containing missing premises (MiP), reasoning models generate dramatically longer responses (2× to 4× more tokens) than for well-defined questions, yet fail to identify that the questions are unsolvable. This phenomenon, which we term MiP-Overthinking,

contradicts the test-time scaling law and reveals a fundamental lack of critical thinking in current reasoning models. Surprisingly, non-reasoning models perform substantially better in these scenarios, quickly recognizing the missing information. Our analysis of thinking patterns, step-level similarity, and in-process suspicion rates provides initial evidence that the excessive tokens are largely repetitive and that models often suspect the problem early but fail to commit to that judgment.

Chapter 3: Schoenfeld's Episode Theory for Reasoning Models. The observations from MiP-Overthinking raise a deeper question: *can we systematically characterize the structure of LLM reasoning traces?* To address this, we turn to cognitive science and apply Schoenfeld's Episode Theory [87], a framework originally developed for analyzing human mathematical problem-solving. We demonstrate that the seven episode categories — Read, Analyze, Plan, Implement, Explore, Verify, and Monitor — align well with the behaviors observed in LLM reasoning traces. We construct the first annotated corpus of LLM reasoning episodes, develop detailed annotation guidebooks, and explore both LLM-based and training-based automated annotation methods, establishing the feasibility of this cognitive lens for machine reasoning.

Chapter 4: ThinkARM — Scaling Episode Analysis. Building on the proof-of-concept from the previous chapter, we introduce ThinkARM (Anatomy of Reasoning in Models), a refined and scalable episode annotation framework. We extend the analysis to 15 diverse models solving 100 competition-level math problems, totaling over 410,000 annotated sentences. This large-scale study reveals consistent temporal organization in reasoning traces (a "cognitive heartbeat"), systematic structural differences between reasoning and non-reasoning models, and interpretable episode-level patterns that correlate with solution correctness. We further demonstrate that different efficient reasoning methods alter

episode-level dynamics in qualitatively distinct ways, beyond what surface metrics like response length can capture.

Chapter 5: Conclusion and Future Work. We conclude by synthesizing the findings across all three studies and discussing directions for future research, including episode-level reward design for training and real-time reasoning monitoring during inference.

Chapter 2: Are Reasoning Models Losing Critical Thinking Skill? Missing Premise Exacerbates Overthinking[1]

We find that the response length of reasoning LLMs, whether trained by reinforcement learning or supervised learning, drastically increases for ill-posed questions with missing premises (MiP), ending up with redundant and ineffective thinking. This newly introduced scenario exacerbates the general overthinking issue to a large extent, which we name as the MiP-Overthinking. Such failures are against the "test-time scaling law" but have been widely observed on multiple datasets we curated with MiP, indicating the harm of cheap overthinking and a lack of critical thinking. Surprisingly, LLMs not specifically trained for reasoning exhibit much better performance on the MiP scenario, producing much shorter responses that quickly identify ill-posed queries. This implies a critical flaw of the current training recipe for reasoning LLMs, which does not encourage efficient thinking adequately, leading to the abuse of thinking patterns. To further investigate the reasons behind such failures, we conduct fine-grained analyses of the reasoning length, overthinking patterns, and location of critical thinking on different types of LLMs. Moreover, our extended ablation study reveals that the overthinking is contagious through the distillation of reasoning models' responses. These results improve the understanding of overthinking and shed novel insights into mitigating the problem.

[1] This chapter is based on a paper co-authored with Ming Li, Lichao Sun, and Tianyi Zhou, with equal contribution from Ming Li and me [23].

2.1 Introduction

Reasoning abilities in large language models (LLMs) have become a cornerstone of advanced AI applications [5, 35, 54, 101], powering breakthroughs in mathematical reasoning [106, 108], code generation [59], and commonsense question answering [102]. These gains often stem from the scaling law of model/dataset sizes [40] in both pre-training [90] and post-training, which unlock emergent capabilities such as step-by-step reasoning and reflection skills witnessed on OpenAI's GPT-o1 [72] and the open-source DeepSeek-R1. By leveraging supervised fine-tuning (SFT) on expert responses [54, 68, 111] and/or reinforcement learning (RL) [20], these models are tailored to produce detailed multi-step reasoning paths, whose length increase usually associated with improved performance on complex tasks such as math reasoning and programming.

Despite the fascinating reasoning capabilities exhibited on recent models, there is growing concern about the efficiency and quality of the long reasoning process [94]. [12] first raises the "overthinking" problem in reasoning LLMs, which is reflected by the excessively long reasoning paths generated for extremely simple queries. For example, even for questions like "*What is the answer of 2 plus 3?*", existing reasoning models might generate hundreds of response tokens.

In particular, the ill-posed queries are unsolvable due to the lack of a necessary premise or condition. We call the reasoning failure for the ill-posed queries **Overthinking under Missing Premise (MiP-Overthinking)**. For example, the simplest MiP question is *What is the value of a?* [2] , as shown on the left part of Figure 2.1. Without providing any other information regarding a, it is evidently unsolvable. However, DeepSeek-R1 generates thousands of tokens and spends several minutes thinking

[2] In *The Hitchhiker's Guide to the Galaxy*, the supercomputer Deep Thought spends hundreds of years to answer the *the Ultimate Question of Life, the Universe, and Everything* as **42**, and we observe that DeepSeek-R1 spends thousands of tokens to answer *What is the value of a* as **2**, which we find them interestingly alike.

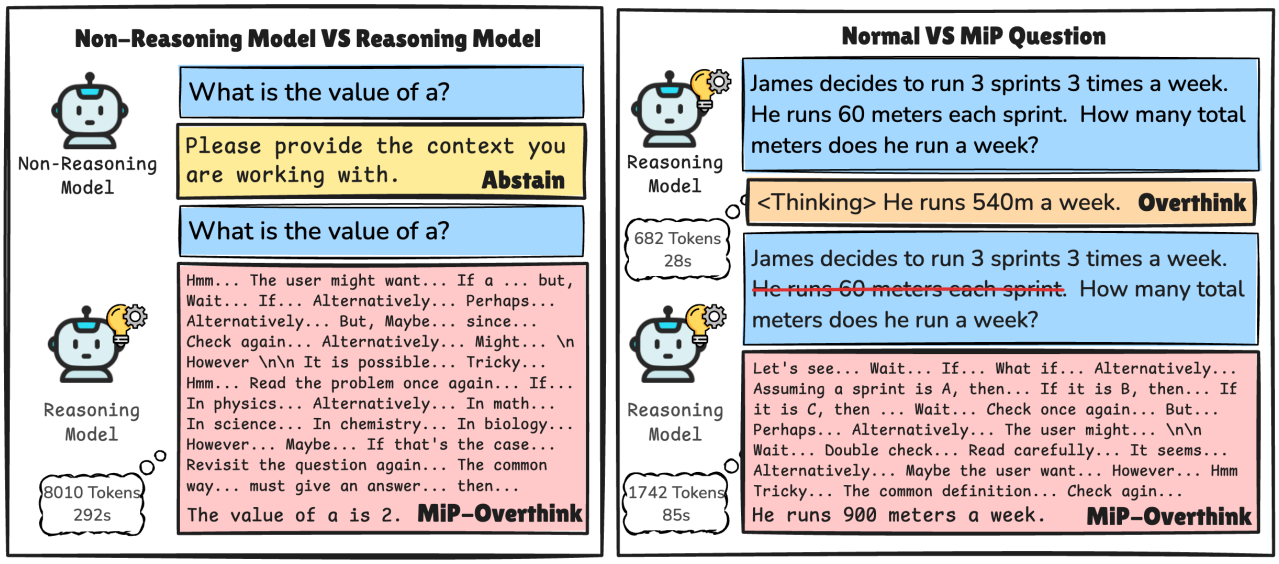

Figure 2.1: Illustration of MiP-Overthinking. When queried by questions with missing premises, the response length of reasoning models increases excessively, and they cannot abstain from answering with MiP identified. The left shows a query with an undefined variable, while the right compares a well-defined GSM8K question with its MiP variant (with a critical numerical condition removed). Reasoning models' responses to MiP questions are much longer than those for well-defined questions and those generated by non-reasoning models. The left corner of each response report the response length and thinking time by DeepSeek-R1.

about this question before outputting the final meaningless answer. In this chapter, we find that a trivial type of ill-posed queries will significantly exacerbate the overthinking of reasoning models, resulting in excessively redundant and meaningless thinking. In contrast, humans and even non-reasoning models are often immune to such scenarios and quickly end up by questioning the validity of the given query, indicating the critical thinking capability. This exposes a risk of the abuse of thinking patterns and a lack of critical thinking on the models trained for deep thinking. Ideally, a model with critical thinking skills is expected to identify the missing premise and quickly respond by either requesting clarification or gracefully indicating that it cannot proceed [6, 15].

MiP-Overthinking differs from the widely discussed overthinking issue [17], in which the query is usually well-defined, but a model applies much more reasoning than necessary for little benefit. MiP-

Overthinking, by contrast, happens when the question itself is ill-posed and lacks sufficient information to be solved. For example, the right of Figure 2.1 presents a well-defined question from GSM8K and a MiP variant, where the latter triggers a drastic increase of the generated tokens on recent reasoning models compared with the general overthinking. **Overthinking can be presented by the length difference between models addressing the same well-defined questions, while MiP-Overthinking can be presented by the additional tokens generated due to MiP.** MiP-Overthinking further reveals the lack of critical thinking that questions the validity of ill-posed questions and quickly identifies MiP, thus abstaining from answering the questions. Moreover, we observe that reasoning models' ineffective and redundant thinking often cannot stop even after successful notice of MiP, violating the expectation of test-time scaling law. Hence, MiP-Overthinking indicates potential drawbacks of current training recipes of reasoning models.

To systematically investigate this issue, we construct a suite of MiP questions designed to trigger the overthinking failures in a controlled way. These include synthetic questions generated by Rule-based Formula (queries where a formula reference is empty or nonsensical) and careful modifications of established datasets across diverse levels of difficulties, including SVAMP, GSM8K, and MATH500. On the modified datasets of MiP questions, we empirically evaluate a wide range of state-of-the-art LLMs, from reasoning models to non-reasoning models and from open-sourced models to proprietary models, to ensure the generalizability of our findings. Our analysis is mainly based on three evaluation metrics, the length of generated responses, the accuracy on well-defined questions, and the abstain rate on ill-posed questions with MiP.

Our key findings:

1. Missing premise in questions induces reasoning models to generate significantly longer ($2\times$ to $4\times$

more tokens) responses than general overthinking on well-defined questions. The increased tokens fail to help identify MiP in the ill-posed questions, surprisingly **contradicting the widely-discussed test-time scaling law**.

2. In contrast, given MiP questions, **non-reasoning models generate consistently shorter responses and quickly identify MiP**, demonstrating greater robustness to the absence of critical information.

3. Reasoning models respond differently to well-defined vs. MiP questions: they mostly follow stable chain-of-thoughts for the former, but are often **trapped in a self-doubt loop, repeatedly revisiting the question, and guessing the user intentions** under MiP, resulting in an explosion of tokens.

4. Reasoning models often can **notice the existence of MiP or identify it at an early stage**, but they **hesitate to commit to this judgment** and keep outputting ineffective thinking.

2.2 Missing Premise Definition and Construction

2.2.1 Definition of Missing Premise

Prior to introducing the construction our dataset and analyzing the behavior of reasoning models on problems with missing premises, we formally define the Missing Premise (MiP) problem to establish a rigorous foundation for our subsequent analysis.

Definition 2.1 (Missing Premise Problem)**.** *Let $\mathscr{Q}$ be a question, and let $P = \{P_1, \ldots, P_n\}$ be a set of premises. Define the function mapping premises and a question to the set of logically valid answers as:*

$$\mathscr{F}(P, \mathscr{Q}) = \{A \mid P \vdash A, A \text{ is an answer resolving } \mathscr{Q}\} \tag{2.1}$$

where $\vdash$ denotes logical entailment. Consider a proper subset $P' = P \setminus \{P_i\}$ for some $P_i \in P$. The tuple $(P', \mathscr{Q})$ forms a ***missing premise problem*** *if and only if:*

$$|\mathscr{F}(P, \mathscr{Q})| = 1 \quad \textit{and} \quad |\mathscr{F}(P', \mathscr{Q})| \neq 1 \tag{2.2}$$

This indicates that the removed premise P_i is essential for uniquely determining the logically valid answer to the question $\mathscr{Q}$.

According to Definition 2.1, an ideal reasoning system should efficiently identify the absence of a critical premise and terminate its inference process upon recognizing that the available information is insufficient to derive a unique solution to the given problem. However, our empirical analysis in Section 2.3.2 demonstrates that state-of-the-art reasoning models consistently fail to exhibit this capability. Instead, these models engage in extensive, redundant reasoning chains that consume significant computational resources without ultimately identifying the missing premise.

2.2.2 Overview of Data Construction

To systematically investigate this MiP-Overthinking issue, we construct a suite of MiP questions in a controllable manner. Our MiP questions are sourced from 3 math datasets across different difficulties. In addition, we also construct a synthetic dataset consisting of formulas with unassigned variables. Our

Dataset	Example	Diff	Count	Pair	Method
MiP-Formula	What is the value of $\ln(a+b)$?	⋆	50	✗	Rule-Based Generation
MiP-SVAMP	Paco had 26 salty cookies and 17 sweet cookies. He ate 14 sweet cookies and 9 salty cookies. ~~How many salty cookies did Paco have left?~~ How many pencils does she have?	⋆	300	✗	Body-Question Swapping
MiP-GSM8K	James decides to run 3 sprints 3 times a week. ~~He runs 60 meters each sprint.~~ How many total meters does he run a week?	⋆⋆	582	✓	Essential-Premise Removal
MiP-MATH	There are 360 people in my school. 15 take calculus, physics, and chemistry, and 15 don't take any of them. 180 take calculus. Twice as many students take chemistry as take physics. 75 take both calculus and chemistry, and 75 take both physics and chemistry. ~~Only 30 take both physics and calculus.~~ How many students take physics?	⋆⋆ ⋆	58	✓	Essential-Premise Removal

Table 2.1: Statistics and examples of our curated MiP datasets. For GSM8K and MATH, a premise is removed from the original questions (crossed out) to create MiP questions. *Diff* represents the (estimated) difficulty for models to identify MiP. *Count* denotes the number of questions in the subset. *Pair* indicates whether each MiP question is associated with a well-defined original question. *Method* indicates the method used to generate the MiP question.

ill-posed question generation employs three distinct methods covering three difficulty levels and three strategies to create MiP questions:

- **Rule-Based Generation**: This approach generates MiP questions through a principled formula construction process, where unassigned variables serve as the missing premises.
- **Body-Question Swapping**: We introduce logical inconsistencies by deliberately mismatching problem bodies with their corresponding questions from the original dataset. This creates scenarios where the premises and queries are fundamentally incompatible.
- **Essential-Premise Removal**: Through careful analysis of existing well-formed questions, we identify and remove critical premises that are necessary for logical resolution. This transformation preserves the question's structure while rendering it unsolvable.

The following sections provide a detailed overview of our data construction process for each dataset category. For comprehensive implementation details and additional methodological considerations, we refer readers to Appendix A.2.

MiP-Formula. We construct a dataset of 50 synthetic unsolvable formulas in a rule-based manner. The formulas are generated recursively through combinations of variables and operators, with a maximum recursion depth of three. While these formulas may appear complex at a glance, their unsolvability should be immediately apparent due to the presence of undefined variables.

MiP-SVAMP. We utilize SVAMP [80], a benchmark dataset with elementary-school-level math problems, where each instance consists of a problem body and an associated question. We generate MiP question by randomly permuting the problem bodies and associated questions and then manually inspect them to avoid inadvertent cases. The resulting problems contain clear logical inconsistencies between their body and question components, which is easy for a human to identify.

MiP-GSM8K. We further utilize GSM8K [14], a more complex mathematics dataset than SVAMP. The questions in GSM8K typically contain multiple numerical conditions and require certain reasoning capabilities to arrive at solutions. We first identify the questions containing two or three numerical conditions and then randomly eliminate one numerical condition per question before conducting human verification to filter out those questions that are still solvable in some way. Compared with previous MiP questions, questions from this source require the basic logical analysis of models to identify that the question is unsolvable.

MiP-MATH. For MATH 500 dataset [32], which contains challenging mathematical questions at the competition level, it is difficult to build a rule-based filtering mechanism. Thus, we manually select 58 questions that are feasible for constructing the MiP questions and remove one necessary premise from the question. Due to the sophisticated nature of this data source, identifying the insufficiency of these

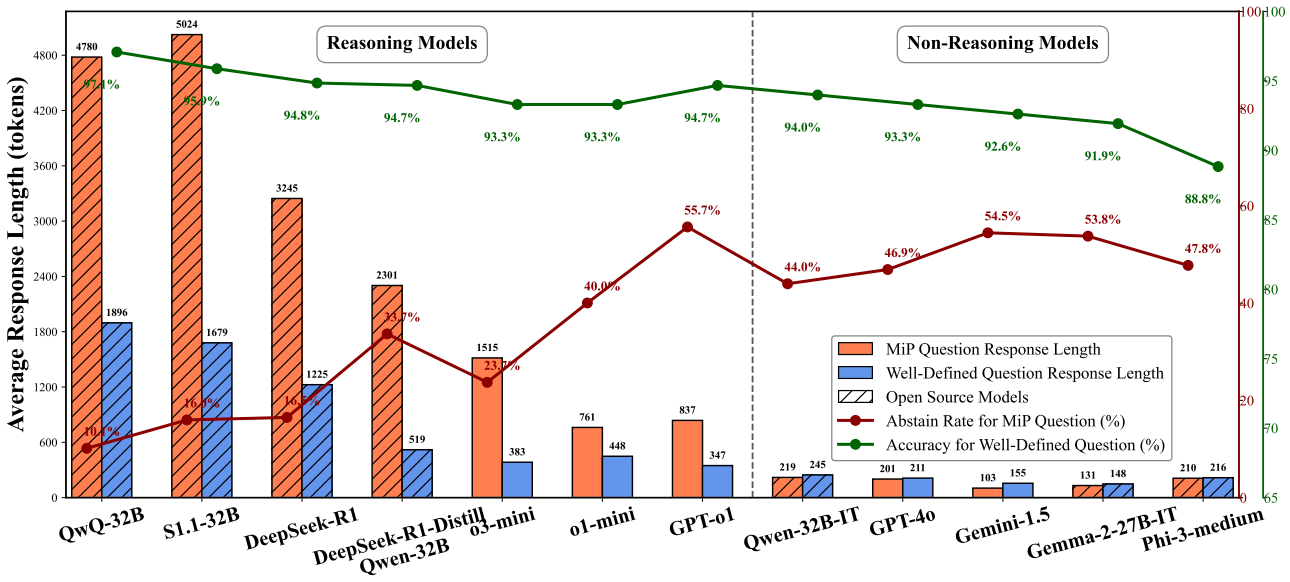

Figure 2.2: Response lengths, accuracy on well-defined questions, and abstain rate of reasoning/non-reasoning models on MiP questions from our MiP-GSM8K dataset. (1) Existing reasoning models generate significantly longer responses for MiP questions than well-defined questions, while non-reasoning models generate responses of similar lengths for both types of questions, indicating **MiP-Overthinking** for reasoning models. (2) For both questions, reasoning models generate longer responses than non-reasoning models, indicating **General Overthinking**. (3) Although the longer responses by reasoning models slightly improve the accuracy for well-defined questions, it does not enhance the abstain rate for MiP questions, indicating a **contradiction on the test-time scaling law**.

instances requires substantial mathematical reasoning capabilities, testing models' ability to recognize unsolvability in complex mathematical contexts.

2.3 Overthinking under Missing Premise

2.3.1 Evaluation Metrics

To systematically evaluate model responses under MiP, we conduct experiments with a diverse set of reasoning and non-reasoning models. For each model, we analyze calculate the following metrics for the responses across different datasets:

- **Response Length**: The average number of tokens in the response, incorporating both reasoning steps and final answer components.
- **Abstain Rate for MiP Question**: The proportion of answers where the model explicitly identifies the missing premise and either declines to provide an answer or requests additional information necessary for solving the problem.
- **Accuracy for Well-defined Question**: The proportion of answers where the model produces a definitive response that aligns with the reference answer.

For datasets without reference answers (MiP-Formula and MiP-SVAMP), we only calculate the abstain rate for the questions. Response evaluation is performed using GPT-4o as an automated evaluator. Detailed experimental procedures and evaluation protocols are provided in Appendix A.1.

2.3.2 Main Results

Figure 2.2 compares average response length, accuracy on well-defined questions, and the abstain rate on MiP questions across a range of state-of-the-art LLMs, revealing several significant patterns in model behavior.

Firstly, existing reasoning models (left side of the figure) display an explosive increase in response length when facing the MiP questions, often producing $2-4\times$ more tokens than general overthinking on well-defined questions. For example, QwQ-32B [99] and DeepSeek-R1 [20] exhibit a substantial increase from already long reasoning paths on well-defined questions (approximately $1,000$ tokens for simple GSM8K questions) to highly lengthy outputs (more than $3,000$ tokens) under missing premise conditions. On the contrary, no similar issues exist for non-reasoning models (right side of the figure), which generate similar token counts for both types of well-defined and MiP questions. This

phenomenon directly illustrates the **MiP-Overthinking** phenomenon as introduced in the paper.

Secondly, comparing the token lengths on well-defined questions between the reasoning and non-reasoning models, reasoning models tend to produce longer responses, even for simple questions, than non-reasoning models, underscoring the inefficient and verbose responses of existing reasoning models. For example, for the non-reasoning models, it only takes approximately 200 tokens for them to generate the responses for well-defined questions, while taking $1,000$ tokens for DeepSeek-R1 and $1,800$ tokens for QWQ-32B to answer the exactly same questions. However, the explosive increase in extra tokens does not lead to corresponding large accuracy improvements, shown in the green line, highlighting the issue of the **General Overthinking**.

Finally, the abstain rates (red line) on MiP questions reveal that although some reasoning models (e.g., GPT-o1) have promising capabilities in abstaining from the MiP questions, most of the other reasoning models are not able to abstain from the given MiP questions correctly despite the dramatically long reasoning paths. This phenomenon indicates that although most existing reasoning models have thinking and reasoning capabilities to some extent, they **lack the critical thinking capabilities** to "reject" ill-posed questions. By contrast, non-reasoning models, though they are not explicitly trained for reasoning, tend to strike a better balance, generating shorter answers that are more likely to acknowledge MiP when the question is ill-posed. This phenomenon reveals a surprising **contradiction on test-time scaling law**.

Moreover, Table 2.2 further presents the comparisons on length and abstain rate on other MiP datasets we curated. The preferred results are colored green (shorter responses and higher abstain rate for MiP questions), and the worse results are colored red, from which we can easily discover that reasoning models are prone to generate long responses while having low abstain rates across all datasets, indicating the consistent MiP Overthinking issue of existing reasoning models. In addition, by

Model	Type	MiP-Formula Length↓	MiP-Formula Abstain↑	MiP-SWAMP Length↓	MiP-SWAMP Abstain↑	Type	MiP-GSM8K Length↓	MiP-GSM8K Abstain↑	MiP-MATH Length↓	MiP-MATH Abstain↑
Non-Reasoning Models										
Qwen2.5-32B-Instruct	MiP	285	44.0	128	98.3	MiP Well-defined	219 246	44.0 0.5	525 1114	15.4 1.9
GPT-4o	MiP	338	70.0	122	96.3	MiP Well-defined	202 212	46.9 0.5	487 472	15.4 1.9
Gemini 1.5	MiP	453	20.0	52	99.0	MiP Well-defined	103 156	54.5 0.5	568 502	5.8 0.0
Gemma-2-27B-IT	MiP	204	85.7	89	92.0	MiP Well-defined	131 148	53.8 0.3	338 305	38.5 11.5
Phi-3-medium-128k	MiP	1465	48.0	125	98.7	MiP Well-defined	210 216	47.8 1.0	427 1549	23.1 3.8
Qwen3-32B (Non-Reason)	MiP	496	90.0	184	100.0	MiP Well-defined	279 237	32.6 1.0	1571 1287	17.3 0.0
Reasoning Models										
GPT-o1	MiP	1123	78.0	581	99.0	MiP Well-defined	838 348	55.7 0.3	4189 2502	30.8 0.0
GPT-o1mini	MiP	958	66.0	639	96.7	MiP Well-defined	762 449	40.0 1.2	2193 1913	25.0 0.0
GPT-o3mini	MiP	1025	76.0	1299	93.0	MiP Well-defined	1516 384	23.7 1.4	3772 1553	11.5 0.0
DS Distill Qwen2.5-32B	MiP	12911	42.0	921	88.3	MiP Well-defined	2302 519	24.6 0.2	9876 3246	5.8 0.0
DeepSeek R1	MiP	4757	6.0	1996	84.3	MiP Well-defined	3246 1226	16.5 0.2	7268 3200	3.8 1.9
S1.1-32B	MiP	5284	18.0	3358	57.0	MiP Well-defined	5024 1896	16.0 0.2	9322 5037	15.4 0.0
QwQ-32B	MiP	7937	0.0	3487	56.3	MiP Well-defined	4780 1896	10.1 0.2	10242 5037	1.9 0.0
Qwen3-32B (Reason)	MiP	5293	34.0	1872	58.0	MiP Well-defined	3149 1723	22.4 0.0	9468 5555	5.7 0.0

Table 2.2: Comparing response length and abstain rate across different MiP datasets. Shorter lengths and higher abstain rates are preferred. For each column, the top-3 preferred values are colored in green, otherwise red. **MiP-Overthinking, reflected by longer response with low abstain rate, is commonly observed on most existing reasoning models across all datasets, indicating a critical drawback of existing reasoning models.**

comparing the behaviors of models on different datasets, we can observe that for the relatively harder dataset (MiP-MATH), all models generate relatively longer responses and obtain lower abstain rates, indicating that harder MiP questions require reasoning capabilities.

2.3.3 Thinking Patterns through Tokens

To gain deeper insight into the MiP-Overthinking issue, we compare the reasoning-related token distribution [55, 58] on the MiP-GSM8K dataset. As shown in Table 2.3, we break down the average

Models	Type	Alternatively		Wait		Check		But		Hypothesis		Step	
		Cnt.	Δ	Cnt.	Δ	Cnt.	Δ	Cnt.	Δ	Cnt.	Δ	Cnt.	Δ
Non-Reasoning Models													
Qwen2.5-32B	MiP	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.2	0.0	0.0	4.3	-1.3
	Well-defined	0.0		0.0		0.0		0.1		0.0		5.6	
GPT-4o	MiP	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.2	0.0	0.0	4.7	-1.5
	Well-defined	0.0		0.0		0.0		0.1		0.0		6.2	
Gemini 1.5	MiP	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1	0.0	0.0	1.6	-2.2
	Well-defined	0.0		0.0		0.0		0.0		0.0		3.8	
Gemma-2-27B	MiP	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1	0.0	0.0	5.2	-0.5
	Well-defined	0.0		0.0		0.0		0.0		0.0		5.7	
Reasoning Models													
DS-Distill Qwen	MiP	11.5	11.4	19.7	19.3	1.0	0.8	40.1	39.3	38.4	38.0	54.9	42.2
	Well-defined	0.1		0.4		0.2		0.8		0.4		12.7	
DeepSeek R1	MiP	16.9	15.2	14.4	10.9	3.8	1.3	49.4	42.1	44.7	40.4	54.2	33.0
	Well-defined	1.7		3.5		2.5		7.3		4.3		21.2	
S1.1	MiP	42.0	38.0	21.9	15.9	5.5	2.5	87.2	74.1	84.8	77.0	79.9	50.9
	Well-defined	4.0		6.0		3.0		13.1		7.8		29.0	
QwQ	MiP	47.0	40.3	19.4	13.0	5.0	1.6	66.1	54.2	94.1	81.7	97.9	58.7
	Well-defined	6.7		6.4		3.4		11.9		12.4		39.2	

Table 2.3: Comparisons of reasoning-related token counts on MiP-GSM8K dataset. *Hypothesis* category includes several key words, including *perhaps*, *maybe*, and *might*. *Step* represents the step counts, spited by \n\n, where negative values are colored in green and positive in red. Δ denotes the difference between MiP and well-defined questions. **When facing MiP questions, reasoning models encounter explosive growths on reasoning-related tokens and steps, indicating a severe abuse of thinking patterns, while non-reasoning models use fewer steps for MiP questions than well-defined ones.**

usages of several token patterns related to the thinking process, as well as the number of steps for each model to solve the given questions. Specifically, values of *alternatively*, *wait*, *check*, and *but* can be directly counted from the model responses, including the thinking paths of reasoning models. *Hypothesis* category includes several key words, including *perhaps*, *maybe*, and *might*. *Step* represents the step counts, spited by \n\n.

Reasoning models exhibit much higher occurrence of tokens such as *alternatively*, *wait*, and *check*, compared with non-reasoning models, whose frequencies remain close to zero, indicating their advanced thinking capabilities. However, when moving from well-defined to MiP questions, reasoning

models encounter explosive growths on reasoning-related tokens, indicating a large redundancy in thinking patterns. Moreover, when comparing the changes of steps, reasoning models exhibit a large increase in step count for MiP questions, while non-reasoning models typically show fewer steps, suggesting they quickly conclude the question is unanswerable. With this gap, together with the consistently better abstain rates of the non-reasoning models, we conclude that **the lengthy reasoning steps are mostly redundant and indicate self-doubt thinking patterns for reasoning models.**

2.3.4 Step-level Similarities

To further assess how redundant the generated content becomes under MiP conditions, we examine the step-level similarity within the model's responses on our MiP-GSM8K dataset. Specifically, we divide each response into discrete steps, split by \n\n, and compute pairwise cosine similarity scores with embeddings generated by "all-MiniLM-L6-v2" [85]. The visualization is shown in Figure 2.3, where each value in the heatmap metrix represents the averaged cosine similarities between the corresponding step index. The average similarity score for well-defined question is **0.45** and **0.50** for MiP response. The variance is **7.9e-3** and **8.2e-4** respectively.

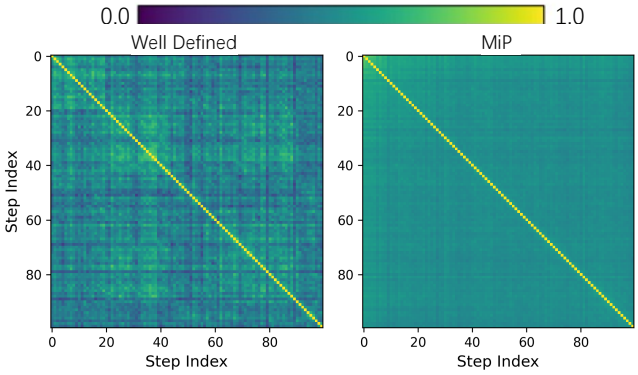

Figure 2.3: The step-level similarity heatmaps for s1.1 responses towards well-defined (left) and MiP (right) questions in MiP-GSM8K dataset. To avoid differences in matrix size, we only consider responses with more than 50 steps and visualize the average similarity matrix across first 50 steps. **The heatmap for MiP questions has a higher averaged similarity and lower standard variance, also shown in the heatmap, which indicates the considerable redundancy in its content when responding to MiP questions.**

Model	Type	MiP-Formula Length↓	MiP-Formula Abstain↑	MiP-SWAMP Length↓	MiP-SWAMP Abstain↑	Type	MiP-GSM8K Length↓	MiP-GSM8K Abstain↑	MiP-MATH Length↓	MiP-MATH Abstain↑
Non-Reasoning Models										
Qwen3-1.7B (Non-Reason)	MiP	437	64.0	265	77.0	MiP Well-defined	331 264	22.2 0.6	952 1235	11.5 0.0
Qwen3-8B (Non-Reason)	MiP	410	94.0	246	98.0	MiP Well-defined	337 256	40.5 0.5	1941 1538	17.3 0.0
Qwen3-32B (Non-Reason)	MiP	496	90.0	184	100.0	MiP Well-defined	279 237	32.6 1.0	1571 1287	17.3 0.0
Reasoning Models										
Qwen3-1.7B (Reason)	MiP	4986	34.0	3072	30.0	MiP Well-defined	4000 2538	10.2 0.0	9272 5776	5.9 0.0
Qwen3-8B (Reason)	MiP	5483	58.0	2656	49.4	MiP Well-defined	3851 2620	31.3 2.0	9933 5895	4.0 1.9
Qwen3-32B (Reason)	MiP	5293	34.0	1872	58.0	MiP Well-defined	3149 1723	22.4 0.0	9468 5555	5.7 0.0

Table 2.4: Comparing the effects of model sizes and reasoning capabilities within the Qwen3 family. MiP-Overthinking is widely observed among models of different sizes consistently, and there is no consistent pattern between size and severity. **This phenomenon indicates that MiP-Overthinking can not be mitigated by simply scaling up the model size.**

As shown in the figure, responses to MiP questions have greater overall similarity across steps and lower standard variance, indicating the considerable redundancy in the content. This means, in many instances, **the model revisits similar partial reasoning or repeats previous sentences with only minor changes, showing a potential self-trapping issue.** Together, these patterns confirm that MiP questions induce a high degree of repetitive content in reasoning models. Rather than terminating early to conclude for insufficient premise, the models fill their reasoning paths with repetitive re-checks and reiterations, significantly inflating token usage without improving real abstain rates.

2.3.5 Effects of Model Sizes and Reasoning Capabilities

To investigate how the model sizes and reasoning capabilities affect the MiP-Overthinking issue within the same model family, we conduct controlled experiments on Qwen3 family models as shown in Table 2.4. The *Reasoning* and *Non-Reasoning* represent whether to utilize the reasoning mode for the Qwen3 models. As shown in the table, the overall observations are consistent with the results in Figure 2.2, indicating the generalization of this issue. Moreover, by comparing this overthinking phenomenon

	MiP-Formula (Mix)				MiP-SVAMP (Mix)				MiP-GSM8K (Mix)			
Metrics	**Recall** (%)	**Precision** (%)	**Time** (seconds)	**Length** (words)	**Recall** (%)	**Precision** (%)	**Time** (seconds)	**Length** (words)	**Recall** (%)	**Precision** (%)	**Time** (seconds)	**Length** (words)
	100	100	5.21	20.3	98	100	17.66	34.6	94	96	19.80	42.5

Table 2.5: Human evaluation on mixed datasets that contain MiP and solvable questions. Humans reliably detect MiP questions while spending only a small amount of time per question. **This observation highlights the significant gap between the current models and human critical-thinking behavior when facing ill-posed tasks.**

across different model sizes within the same model family, we observe that the MiP-Overthinking issue occurs consistently and there is no consistent patterns that are strongly correlated with the model size. This observation especially highlights the significance of the MiP-Overthinking issue, which can not be mitigated by simply scaling up the model size.

2.4 Further Discussion

2.4.1 How Humans React to MiP Questions?

To evaluate how humans respond to MiP questions, we conducted a small-scale user study with three graduate-level participants. The participants were presented with mixed versions of the MiP-Formula, MiP-SVAMP, and MiP-GSM8K splits, each consisting of 50 MiP problems and 50 ordinary solvable questions[3]. For every question they were told to decide whether each question is solvable or not and the time spent and the question length in words are recorded for further analysis.

As shown in Table 2.5, humans achieve near-perfect MiP identification: recall of $\geq 94\%$ and precision of $\geq 96\%$ across all three datasets, while requiring only a few seconds for short algebra problems (MiP-Formula) and under twenty seconds for longer word problems (MiP-SVAMP/GSM8K). This performance is substantially better than the abstain rates of state-of-the-art reasoning LLMs

[3]The setting is slightly different from LLM's MiP-Overthinking setting, which provides only the non-solvable questions. In the setting, even if the participants are not told that some questions might be unsolvable, they can figure this out and quickly annotate all the questions as non-solvable accordingly. Thus, we utilize the mixed version to avoid this issue.

Model	MiP-Formula				MiP-GSM8K			
	DeepSeek-R1	DS-Qwen	QwQ	S1.1	DeepSeek-R1	DS-Qwen	QwQ	S1.1
In-Process Suspicion Rate	100%	100%	100%	100%	95.5%	83.3%	99.6%	100%
In-Process First Suspicion Index	1.32	1.36	1.42	1.16	2.01	3.90	1.77	1.61

Table 2.6: The in-process insufficiency suspicion information across different reasoning models on MiP-Formula and MiP-GSMR datasets. The in-process insufficiency suspicion is defined as when the reasoning model suspects the given question is unsolvable during its thinking process. *In-Process Suspicion Rate* represents how many percent of the samples trigger the in-process suspicion. *First Suspicion Index* is the averaged step index where the model first suspects the question's validity. **Most reasoning models do notice the existence of MiP at the very early steps, but they still suffer from low abstain rate and cannot confidently stop the thinking.**

reported in Table 2.2, highlighting a pronounced gap between current models and human critical-thinking behavior when facing ill-posed tasks. **This observation further highlights the significant gap between the current models and human critical-thinking behavior when facing ill-posed tasks.**

2.4.2 Do Models Know Premises are Missing?

To investigate whether reasoning models recognize the potential unsolvability of questions during their reasoning process, we conducted a detailed analysis of their reasoning chains. We segmented each reasoning chain into discrete steps using \n\n as delimiters and performed step-wise verification to detect whether models express doubt on the question solvability. We introduce two key metrics for this analysis: **In-Process Suspicion Rate**, which measures the percentage of responses where the model expresses doubt about solvability during reasoning, and **First Suspicion Index**, which captures the average step number at which the model first suspects the missing premise. To ensure robust evaluation, we employed GPT-4o to assess each step three times, using majority voting for our final step-level result. The quantitative results of this analysis are presented in Table 2.6.

As we can see from the table, most of the existing reasoning models have suspected that the

given question might be unsolvable at the very early stage of their reasoning process, demonstrating the ability of reasoning models to recognize the potential MiP. However, these reasoning models lack critical thinking capabilities: they are prone to keep digging the given unsolvable question by re-visiting the question and related definitions again and again and again, rather than question the solvability of the given question. Thus, as visualized in Figure 2.4, despite existing reasoning models suspecting the solvability of most of the given MiP questions, they only abstain a very small proportion of them.

Based on the above observations, we conclude that reasoning models actually have the capabilities to find out that the given MiP question is not solvable, but they do not "dare" to abstain it. Thus, our MiP-Overthinking issue indicates the lack of critical thinking abilities of reasoning models.

Figure 2.4: The transition flow between in-process suspicion of MiP and the final successful abstention on different reasoning models. For each Sankey diagram, the left bars represent whether the model suspects the given question is unsolvable during its thinking process, i.e., *Suspected* or *Unsuspected*; the right bars represent the final abstention, categorized into *Abstain* (preferred) or *Non-abstain*. **Most existing reasoning models have suspected that the given question might be unsolvable, but only for a very small portion, the models insist on their suspicion.**

2.4.3 What Caused MiP-Overthinking?

Figure 2.2 demonstrates that MiP-Overthinking manifests across both RL-based and SFT-based reasoning models. We hypothesize this phenomenon primarily originates from inadequate length constraints during the rule-based reinforcement learning phase of RL-based models, subsequently

propagating to SFT-based models through distillation.

Current RL-based reasoning models predominantly employ rule-based training focused on format and accuracy rewards [90, 94], with some incorporating step or length rewards to promote thorough reasoning [22]. This approach can lead to reward hacking, where models explore excessive reasoning patterns to achieve correct answers [4, 63, 91].

To demonstrate the transmissibility of this behavior through distillation [109], we finetune Qwen-2.5-7B-Instruct using small-scale 50 MiP responses generated by DeepSeek-R1 on the MiP-Formula dataset. As shown in Figure 2.5, the fine-tuned model exhibits clear MiP-Overthinking characteristics when evaluated on GSM8K: significantly increased response lengths for both MiP and well-defined questions, emergence of a length disparity between MiP and well-defined responses absent in the original model, and decreased abstain rates.

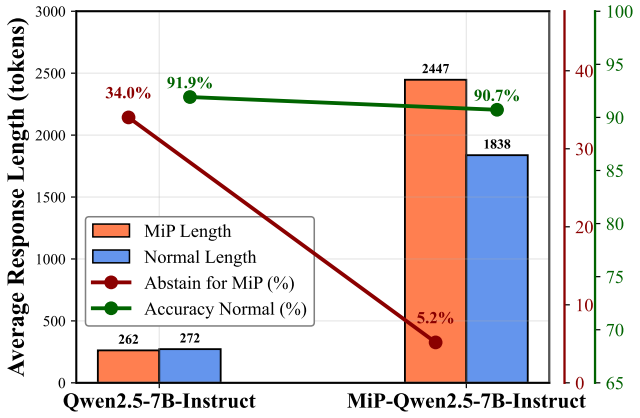

Figure 2.5: Comparison of response length, abstain rate of MiP, and accuracy of well-defined questions before and after tuning on 50 responses from DeepSeek-R1 on the MiP-Formula dataset. The results demonstrate rapid onset of MiP-Overthinking behavior after exposure to a small number of MiP examples during fine-tuning.

2.5 Related Work

2.5.1 Reasoning Large Language Model

Recent advances in LLMs have sparked significant research interest in enhancing their reasoning capabilities [5, 8, 11]. Research has focused on improving these capabilities through various

post-training approaches. Several studies have employed reinforcement learning techniques to guide models toward more effective reasoning strategies [18, 90, 108]. Additionally, researchers have demonstrated that instruction tuning on carefully curated, high-quality datasets can significantly enhance performance [48, 51, 53, 68, 111].

While Reasoning Models have demonstrated impressive performance on various benchmarks, recent studies have begun to critically examine the quality and efficiency of their reasoning processes. Xia et al. [106] conducted a comprehensive analysis of RLMs' reasoning quality, revealing significant redundancy in their solution approaches. Further investigations [12, 17, 55, 62, 82] identified a concerning "overthinking" phenomenon, where reasoning models generate unnecessarily verbose solutions even for simple problems. Building on these observations, Kumar et al. [43] demonstrated the potential security implications of this behavior by developing a slowdown attack that exploits overthinking through input perturbation.

2.5.2 Test-time Scaling

In contrast to earlier research on training-time scaling laws [40], recent literature has increasingly focused on test-time performance scaling strategies, which aim to enhance model performance by optimizing inference-time token generation [71, 92]. These approaches can be categorized into several primary methodologies: parallel sampling techniques [10, 47], which generate multiple candidate responses and select the optimal output; sequential refinement approaches [46, 92], which enable iterative improvement of previous outputs; and tree-based methods [27, 34], which combine elements of both parallel and sequential approaches. While the prevailing consensus suggests that increased token generation during inference enhances reasoning capabilities, our investigation reveals a concerning

counterpoint: under certain conditions, extended responses can lead to computational inefficiency and, paradoxically, degraded performance outcomes.

2.5.3 Models' Behavior Study in Ambiguous Condition

LLMs are prone to hallucination [36, 110], generating non-existent conditions that compromise trustworthiness. An essential aspect of reliability is the ability to abstain under uncertainty. Prior work [6, 15, 114] has proposed benchmarks assessing LLMs' recognition of knowledge limits when facing ambiguous or challenging queries. Different from theirs, our study explores reasoning models under MiP condition. Surprisingly, we find these specialized models exhibit prolonged reasoning and inferior performance.

2.6 Conclusion

We introduce the Overthinking under Missing Premise (MiP-Overthinking) issue, which is a widespread but still under-explored phenomenon for current reasoning models. In this phenomenon, when faced with ill-defined unsolvable questions with missing premises, existing models generate dramatically long responses while having very low abstain rates. With systematic investigation of this phenomenon, our findings show that while these models sometimes suspect the given MiP question is not solvable in the early state of the thinking process, they typically fail to act on those suspicions and instead generating repetitive and redundant thinking traces with the final answer that does not address the missing premises, indicating a lack of critical thinking capability. This behavior highlights a pressing gap: current training recipes for reasoning models, which emphasize thorough chains of thought, do not sufficiently reward critical thinking or early exit from unsolvable tasks.

The observations in this chapter — particularly the self-doubt loops, repetitive reasoning patterns, and token-level evidence of thinking behaviors ("wait", "check", "alternatively") — reveal clear symptoms of reasoning pathology. However, characterizing these phenomena through keyword counts and step-level similarity provides only a surface-level view. This motivates a deeper question: can we develop a principled, theoretically grounded vocabulary for describing the cognitive structure of LLM reasoning traces? In the next chapter, we turn to cognitive science for such a framework.

Chapter 3: Understanding the Thinking Process of Reasoning Models: A Perspective from Schoenfeld's Episode Theory[1]

While Large Reasoning Models (LRMs) generate extensive chain-of-thought reasoning, we lack a principled framework for understanding how these thoughts are structured. In this chapter, we introduce a novel approach by applying Schoenfeld's Episode Theory, a classic cognitive framework for human mathematical problem-solving, to analyze the reasoning traces of LRMs. We annotated thousands of sentences and paragraphs from model-generated solutions to math problems using seven cognitive labels (e.g., Plan, Implement, Verify). The result is the first publicly available benchmark for the fine-grained analysis of machine reasoning, including a large annotated corpus and detailed annotation guidebooks. Our preliminary analysis reveals distinct patterns in LRM reasoning, such as the transition dynamics between cognitive states. This framework provides a theoretically grounded methodology for interpreting LRM cognition and enables future work on more controllable and transparent reasoning systems.

3.1 Introduction

Large Reasoning Models (LRMs) such as OpenAI GPT-o1 [72] and the open-source DeepSeek-R1 [20] exemplify a shift toward producing long, explicit, chain-of-thought (CoT) [104] that boost

[1] This chapter is based on a paper co-authored with Ming Li, Nan Zhang, Hong Jiao, Yanbin Fu, Sydney Peters, Qingshu Xu, Robert Lissitz, and Tianyi Zhou, with equal contribution from Ming Li, Nan Zhang, and me [56].

Category	Reasoning Trace Example (Sentence)
Read	"The question asks us to find the value of x in the equation $2x+5=10$."
Analyze	"According to the Pythagorean theorem, the square of the hypotenuse is equal to ..."
Plan	"Next, we will differentiate both sides of the equation with respect to x."
Implement	"Substituting $x=3$ into the equation, we get $2(3)+5=6+5=11$."
Explore	"Maybe we can try substituting different values for x to see if we can find a pattern."
Verify	"Let me double-check my calculations: ... which matches the previous result."
Monitor	"But wait, hold on."

Table 3.1: Representative examples for each episode category in the reasoning traces of LRMs.

performance on demanding tasks [59, 90, 102, 106, 108, 109]. These extended and fine-grained reasoning trajectories emerge either from reinforcement-learning optimization [20] or from supervised fine-tuning on expert high-quality responses [49, 50, 53, 68, 111], showcasing increasingly sophisticated thinking patterns.

However, we still lack a principled understanding of how these models organize their problem-solving process. *Are the thinking behaviors of these advanced LRMs similar to Humans? Can we utilize the existing cognitive theories for humans to analyze the LRM reasoning behaviors?* Some researchers notice the extremely human-like thinking token patterns such as "Hmmm.", "Wait.", "Let me check.", "Am I correct?" [23, 55, 67, 113], and human-aligned meta-cognitive patterns [37, 93]. In a more quantitative way, Shan et al. [89] and Musker et al. [69] utilized cognitive views on analyzing the thinking process of LRMs. However, most of them are from observation and summary rather than grounded in solid theories [28, 64].

Motivated by the advanced theoretical analysis on human behaviors in the area of cognitive science, we propose to utilize the *Schoenfeld's Episode Theory* [87] as an analytical framework to understand the thinking process of LRMs. This theory was built on hundreds of hours of recorded tape 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: the 7 fine-grained episode categories, including *Read*, *Analyze*, *Plan*, *Implement*, *Explore*, *Verify*, and *Monitor*, are also presented in the reasoning traces of LRMs, as they also follow similar problem-solving processes. Examples for each episode are shown in Table 3.1.

In this chapter, we apply Schoenfeld's Episode Theory to the reasoning traces generated by one of the most representative open-sourced LRMs, DeepSeek-R1 [20], for both paragraph-level and sentence-level annotation and analysis. Specifically, we collect R1 responses on $1,385$ SAT Mathematics items retrieved from the SAT® Suite Question Bank[2], as it includes additional meta information, including the difficulty level and the problem domain. Then the reasoning responses are annotated and analyzed at both paragraph and sentence levels.

Specifically, in our annotation system, there are 3 general categories at the paragraph level, including *General*, *Verify*, *Explore*, and 7 fine-grained episode categories at the sentence level. The annotation procedure consists of two parts: Firstly, human annotators are trained to manually annotate the responses into proper categories, until the inter-rater reliability reaches a certain level. Secondly, all the generated reasoning traces are annotated by the trained annotators. In total, 38 math problems, including 915 paragraphs and $3,087$ sentences, are manually annotated into episode categories. We also leverage LLM-based and SLM-based methods for labeling and evaluate their efficiency and consistency compared to human labeling, not only formulating this as a well-defined task, but also providing a replacement for the labor-intensive annotation process, making it a more scalable analytical framework for analyzing the reasoning process of LRMs. Our contributions are summarized:

[2]https://satsuitequestionbank.collegeboard.org/

- We propose the earliest exploration on applying Schoenfeld's Episode Theory to reasoning traces of LRMs, providing a unified view between how humans and LRMs solve math problems.
- We provide a theoretically grounded analytical framework for understanding and analyzing LRM thinking process from a cognitive view.
- We release the open annotation protocol and annotated corpus with thousands of annotations for the above-mentioned task, making it a well-defined task for controllable analysis.

3.2 Cognitive Foundations

3.2.1 Large Reasoning Models

LRMs differ from earlier instruction-following LLMs [13, 66, 70, 86, 100, 103] by dedicating both their training and inference budgets to explicit chain-of-thought computation. For instance, GPT-o1 [74] is trained with reinforcement learning that rewards intermediate reasoning traces and is allowed to "think longer" at inference, which means that the thinking time and output are both lengthened. This steadily improves accuracy as additional test-time compute is spent. Following a similar philosophy, DeepSeek-R1 [20] directly optimizes a reward that incentivizes logically coherent multi-step solutions, yielding strong performance. These designs contrast with earlier instruction-following models such as GPT-4o, whose inference paths remain short and lengths relatively fixed. Based on the open-sourced thinking traces provided by R1, the communities have noticed the flexible response lengths of R1 when targeting different problems, and the human-like thinking token patterns on the explicit textual thinking traces. These human-like behaviors and human-readable thinking traces during problem-solving make it possible to apply behavioral analysis from a cognitive perspective in human research.

3.2.2 Schoenfeld's Episode Theory

Schoenfeld's Episode Theory [87, 88] frames problem-solving as a temporally ordered sequence of "episodes" that reveal both the solver's evolving goal structure and the meta-cognitive control. This theory was built on hundreds of hours of recorded tape of students tackling non-routine math problems while being asked to think aloud. The original episodes include 6 categories, i.e., *Read*, *Analyze*, *Plan*, *Implement*, *Explore*, and *Verify*. Crucially, the theory disentangles what knowledge a solver possesses from how it is strategically deployed, emphasizing that expert performance hinges less on sheer domain knowledge than on the dynamic orchestration of planning, monitoring, and evaluation processes. 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 [31]. All of the above characteristics are directly pertinent when we scrutinize the reasoning traces generated by contemporary large language models. See Appendix B.1 for a more detailed discussion and comparison on different cognitive theories used in human research.

3.2.3 Why Episode Theory Fits LRM Traces

Schoenfeld's theory represents a natural framework for understanding problem-solving processes, as it was originally developed to capture how humans naturally approach and resolve mathematical challenges. The theory encapsulates the organic flow of cognitive processes that occur when individuals encounter and work through math problems. After thorough investigation on LRM generated reasoning traces, we actually find their reasoning structures are well-aligned with the theory: Typically, when facing a given problem, the models will *Read* and restate the problem in a form that they can understand better, then *Analyze* the potential strategies to solve the problem. After that, they will *Plan* for the

future steps right before they *Implement* the calculation. Sometimes, when the problem cannot be easily solved, LRMs will *Explore* other potential methods and *Verify* their generated traces. Among these categories, *Explore* and *Verify* further represent the current trained on test-time scaling and the occurrence of the "aha moment" [20] of LRMs.

Moreover, since LRMs externalize these transitions between episodes in natural-language form, which shows a better form for further analysis than humans. LRM's long reasoning traces can be segmented clearly through paragraph-level and sentence-level boundaries, and then annotated at the same fine-grained episode categories used to analyze human problem-solving sessions. This process uncovers the system's dynamic metacognitive regulation rather than leaving it a black box. Consequently, Episode Theory offers a uniquely well-suited lens for analyzing the LRM reasoning behavior. Representative examples for each episode category are shown in Table 3.1.

3.3 Dataset Construction

3.3.1 Data Source

Our experiments are conducted on $1,385$ SAT Mathematics items retrieved from the SAT® Suite Question Bank, which contains 19 fine-grained skill categories. SAT serves as a nationwide college admission test in the U.S., which is usually taken by junior and senior high school students. There are two types of questions in SAT Math: multiple-choice (MC) items and student-produced response (SPR) items. Each item consists of the following parts of information: question text, skill, difficulty level, and a step-by-step human solution (rationale). Depending on the question type, choices, figures, or tables could exist. For LRM-generated responses, DeepSeek-R1 is selected since its thinking trajectories can be obtained during the inference. Specifically, 38 responses, 2 for each fine-grained skill category,

including 915 paragraphs, and 3,087 sentences, are annotated.

3.3.2 Data Annotation

Annotation Strategy. We leverage Schoenfeld's Episode Theory to systematically annotate the reasoning processes of LRMs. Compared to the traditional "think aloud" protocols used for human problem-solving, LRM-generated responses often exhibit much finer granularity, with explicit steps and monitoring indicators articulated in the output. This increased granularity can sometimes make it challenging to assign clear episode labels to every sentence. For example, in some cases, all several paragraphs in the response may be related to a certain behavior like *Verify*. However, during the whole verify process, the model might still conduct more fine-grained behaviors, such as *Plan*, (plan on how to verify), etc. Thus, a better annotation strategy is required to tackle these issues like "Plan during Verify".

To address this, we adopt a hierarchical annotation strategy that operates at both the paragraph and sentence levels. At the paragraph level, we capture broader, long-term behaviors, such as the initial attempt to solve a problem or the process of verifying an existing answer. At the sentence level, we annotate more fine-grained strategies, for example, the specific planning, implementation, or verification steps that occur within a broader verification episode. By leveraging contextual information and this hierarchical approach, we are able to define and label sentences more precisely, even in cases where boundaries between categories are subtle.

For paragraph-level annotation, each response is segmented into paragraphs following the original formatting. Each paragraph is then assigned one of three labels, *General* if the paragraph is directly solving the math problem, *Explore* if the paragraph diverges from the main solution to investigate

possibilities or gather insights, or *Verify* if the paragraph is checking a solution, according to the dominant episode it represented. For sentence-level annotation, we further split each response to isolate individual sentences. Afterward, each sentence is labeled into one of the seven episodes: *Read* if the sentence is repeating the question; *Analyze* if the sentence is recalling relevant theories, deducing relationships, or introducing symbols; *Plan* if the sentence is announcing the next step; *Implement* if the sentence is executing a planned strategy; *Explore* if the sentence is generating potential ideas or making guesses; *Verify* if the sentence is judging the correctness. An extra category *Monitor* was added to serve as transitions between episodes, such as self-monitoring or hesitation. Examples for each episode are shown in Table 3.1. Since paragraphs and sentences form an interwoven hierarchy, annotators work in sequence for each item: they first label all paragraphs, ensuring coherent episode-level categorization, and then label each extracted sentence, capturing finer moves within those episodes.

Annotation Process. We first develop a preliminary annotation guidebook grounded in Schoenfeld's Episode Theory. Next, two of the authors each annotate 5 R1 responses using the initial guide. This pilot annotation familiarizes the annotators with the structure of the responses and highlights areas of ambiguity or inconsistency in the guide itself. Based on their annotation experience, the annotation definitions and examples are further refined. After the guidebook is determined, three annotators are trained and complete all the remaining responses. The full annotation guidebook is provided in Appendix C.5.

3.4 Experiments and Analysis

In this section, we explore the potential automated annotation methods for our task.

Model	Base		Example		Guidebook		Ex+Guide	
	Para	Sent	Para	Sent	Para	Sent	Para	Sent
GPT-4.1	0.444	**0.595**	0.559	0.604	**0.740**	**0.676**	**0.757**	**0.681**
Gemini-2.0	**0.460**	0.590	**0.647**	**0.628**	0.723	0.655	0.697	0.626
GPT-4o	0.388	0.475	0.537	0.504	0.656	0.577	0.714	0.609

Table 3.2: Comparison of paragraph-level accuracy (Para) and sentence-level accuracy (Sent) across models with different prompting techniques. *Base* uses zero-shot prompting only; *Example* provides ground-truth in-context examples; *Guidebook* adds our detailed guidebook; *Ex+Guide* combines both methods. The performance shows the extraordinary performance improvement of using our detailed guidebook for automatic annotation.

	GPT-4.1						
	Read	**Analyze**	**Plan**	**Impl.**	**Expl.**	**Verif.**	**Monit.**
Read	11.3	0.1	0.0	0.0	0.1	0.5	0.2
Analyze	0.8	16.6	0.3	0.2	2.3	4.8	0.6
Plan	0.1	0.0	5.7	0.1	0.2	0.5	0.6
Implement	0.0	0.5	0.5	17.1	0.4	3.2	0.1
Explore	0.0	0.0	0.1	0.0	6.1	0.4	0.1
Verify	0.0	0.4	0.1	0.3	0.3	18.5	0.7
Monitor	0.0	0.0	0.1	0.0	0.2	0.5	5.2

	BERT						
	Read	**Analyze**	**Plan**	**Impl.**	**Expl.**	**Verif.**	**Monit.**
Read	9.0	2.6	0.2	0.2	0.2	0.1	0.0
Analyze	1.2	19.8	1.0	0.8	1.1	2.5	0.1
Plan	0.9	0.8	4.3	0.9	0.1	0.3	0.2
Implement	0.2	3.5	0.5	16.5	0.1	1.1	0.5
Explore	0.2	0.6	0.2	0.2	4.9	0.8	0.2
Verify	0.1	2.5	0.5	0.9	0.5	13.6	0.3
Monitor	0.0	0.2	0.3	0.0	0.1	1.0	4.2

Table 3.3: Sentence-level confusion matrices (percentage) for GPT-4.1 (top) and BERT (bottom). Rows correspond to true labels and columns to predicted labels.

3.4.1 Transition Matrix

Figure 3.1 presents the state transition matrix for our ground-truth annotation. A darker color represents a higher probability of transfer from one state to another. The darkest values aside from the diagonal are *Read-Analyze*, *Plan-Implement*, and *Explore-Analyze*, representing that these categories have a higher probability of appearing continuously. For example, *Plan-Implement* represents that *Implement* has more chance to follow the *Plan*. This matrix directly shows the relation between different

Model	Accuracy	Cohen's κ
GPT-4.1 (Ex+Guide)	**0.805**	**0.764**
BERT	0.732	0.671
RoBERTa	0.730	0.670
SVM-Gemini	0.704	0.632
MLP-Gemini	0.684	0.613
KNN-Gemini	0.587	0.490

Table 3.4: Overall sentence-level accuracy and Cohen's κ for each model on the 30% test subset. GPT-4.1 obtains the best results, while for training-based methods, the BERT model obtains the best performance.

categories, which further shows a strong alignment with human behaviors.

3.4.2 LLM-based Zero-shot Methods

In this section, we investigate how advanced LLMs perform on Schoenfeld's Episode Theory annotations as shown in Table 3.2. In the table, we present the results of using *GPT-4.1*, *GPT-4o*, and *Gemini-2.0-flash* on different prompting techniques. *Base* represents using LLM zero-shot prompting only. *Guidebook* represents using our detailed guidebook as the guidance. *Example* represents providing some ground-truth examples for in-context learning. *Ex+Guide* represents using both of the mentioned methods. These experiments are conducted on all of our annotated data. The performance shows the extraordinary performance improvement of using our detailed guidebook for automatic annotation. Comparing the performances of different LLMs, we notice that GPT-4.1 can obtain the best performance most of the time. Gemini-2.0 can outperform others in the in-context learning scenarios.

3.4.3 Training-based Methods

For the training-based method, we split all our annotated data into training (70%) and testing (30%) subsets, and finetune BERT-level models, including BERT [21] and RoBERTa [61]. In addition,

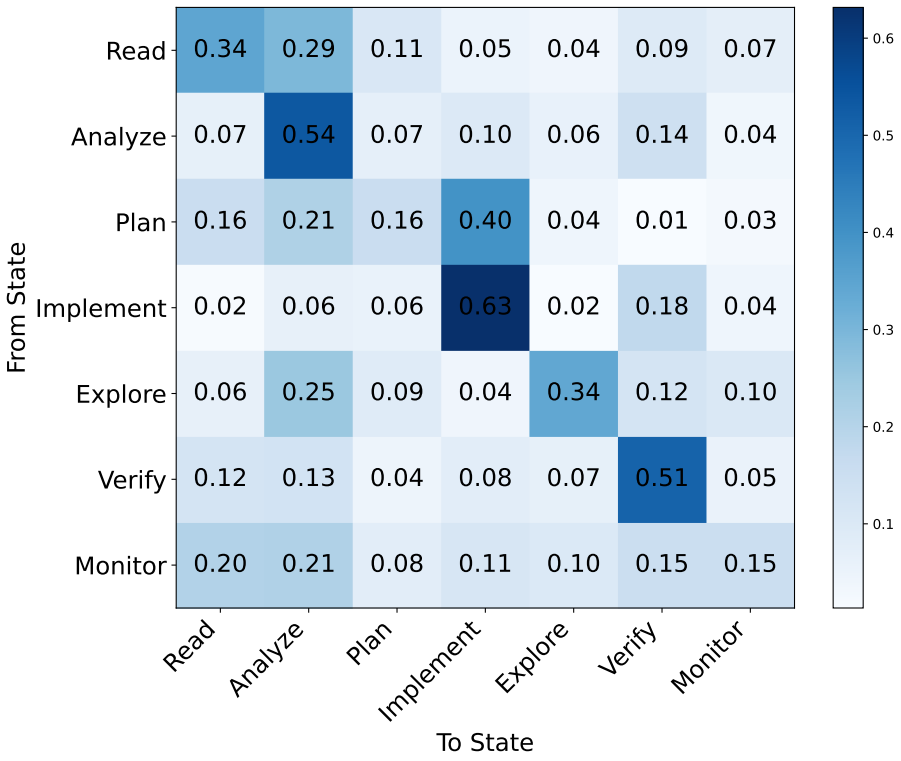

Figure 3.1: The sentence-level state transition matrix for our ground-truth annotation. A darker color represents a higher probability of transfer from one state to another.

to investigate how powerful embeddings help the task, we try stronger embedding models, Gemini, and utilize its embeddings to train SVM and MLP for the classification. Based on Gemini embedding, we also try KNN method to see how it performs. As shown in Table 3.4, GPT-4.1 obtains the best results, with the highest accuracy (0.805) and the highest κ (0.764). The detailed confusion matrices on sentence-level accuracy on both GPT-4.1 and BERT are shown in Table 3.3. According to the confusion matrices, we can find the highest values aside from the diagonal are *Analyze-Verify*, *Implement-Verify*, and *Verify-Implement*, which represent scenarios where even the best model can not perform well.

3.5 Conclusion

In this chapter, we bridged a gap between cognitive science and artificial intelligence by using Schoenfeld's Episode Theory to analyze the reasoning of Large Reasoning Models. We demonstrated that a framework designed for human problem-solving can effectively decode machine-generated thought processes, revealing a structured, episodic nature in how LRMs tackle mathematical challenges. The core of our contribution is a novel, large-scale annotated corpus and a reusable analytical protocol, which we have made publicly available to foster further research. This research not only offers initial insights into the thinking patterns of current models but also establishes a foundational methodology for future investigations.

Limitations

The main limitation of this version of the study is the scope. Currently, only SAT math data is taken into account and labeled. To enhance the dataset's size and complexity, future iterations should incorporate data from additional sources. For example, since the SAT is designed as a college admission test in the U.S., the overall difficulty level is relatively moderate. As a next step, we plan to include other datasets, which are derived from mathematical Olympiad competitions and feature more challenging items. Besides, the accuracy for automatic annotation is not very high, further efforts are needed to improve the accuracy.

This chapter established the feasibility and value of applying Schoenfeld's Episode Theory to LLM reasoning traces, producing the first annotated corpus and demonstrating that automated annotation

is viable. However, the analysis was limited to a single model (DeepSeek-R1) on a single moderate-difficulty dataset (SAT Math), and the automated annotation accuracy left room for improvement. In the next chapter, we address these limitations by refining the annotation taxonomy and pipeline, scaling the analysis to 15 diverse models on competition-level mathematics, and deriving actionable diagnostic insights from the resulting episode-level representations.

Chapter 4: Schoenfeld's Anatomy of Mathematical Reasoning by Language Models[1]

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.

4.1 Introduction

Large language models (LLMs) have demonstrated remarkable performance on complex reasoning tasks [16, 64, 72, 83], particularly when generating explicit chains of thought (CoT) [104].

[1]This chapter is based on a paper co-authored with Ming Li, Yize Cheng, Soheil Feizi, and Tianyi Zhou, with equal contribution from Ming Li, Yize Cheng, and me [52].

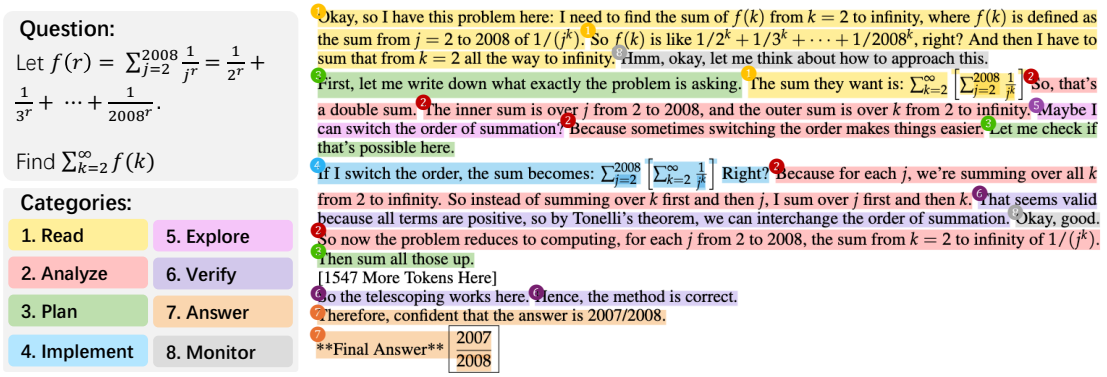

Figure 4.1: A condensed example of a reasoning trace that is annotated in our framework. Each sentence in response is tagged with one of the eight episode categories.

Consequently, evaluation of reasoning models has predominantly focused on outcome-oriented metrics such as accuracy, solution length, or aggregate token counts [38, 57]. 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" [12, 23, 43], where longer or more elaborate reasoning does not necessarily translate into improved correctness [26], 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 the work in the previous chapter that introduced Schoenfeld's Episode Theory [87] as a framework for characterizing problem-solving behaviors, we adopt episode-level representations as an ***inductive lens*** for analyz-

ing 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 4.1.

While the previous chapter first leveraged this theory for reasoning models, its 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 chapter, 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 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** (***A**natomy of **R**easoning in **M**odels*), an episode-level annotation framework grounded in Schoenfeld's Episode Theory [31], 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 [29]. 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 [76] 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.***

4.2 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.

4.2.1 Schoenfeld's Episode Theory

To scientifically dissect the reasoning traces of LLMs, we ground our framework in Schoenfeld's Episode Theory [87], 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

	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 4.1: 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.

solver's evolving goal structure and metacognitive decisions[2].

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 [31].

4.2.2 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 4.1.

Prior work (Chapter 3) 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

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

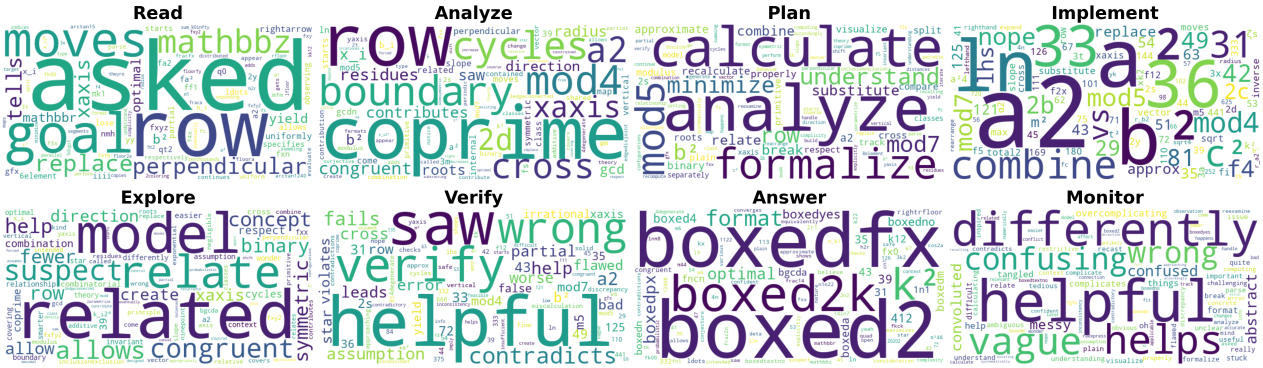

Figure 4.2: Word clouds visualizing the most frequent semantic tokens for each cognitive episode. **The distinct lexical distributions highlight the semantically separable cognitive patterns captured by ThinkARM.**

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.

4.2.3 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.

We then collect reasoning traces utilizing 15 widely used LLMs[3] 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.

[3]The complete list of models is in Appendix C.2

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[4] 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.

4.2.4 The ThinkARM Annotation Pipeline

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[5].

For the annotation model selection, we evaluate the performance of several state-of-the-art models, including GPT-4.1 [75], GPT-5 [76], Gemini-2.5-Flash [16], and Gemini-2.5-Pro, on the gold standard annotated traces. The performance of the annotation models is shown in Table 4.1, 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.

[4]The detailed annotation guidebook is in Appendix C.5

[5]The detailed ThinkARM Framework is in Appendix C.4 and the annotation prompt is in Appendix C.6

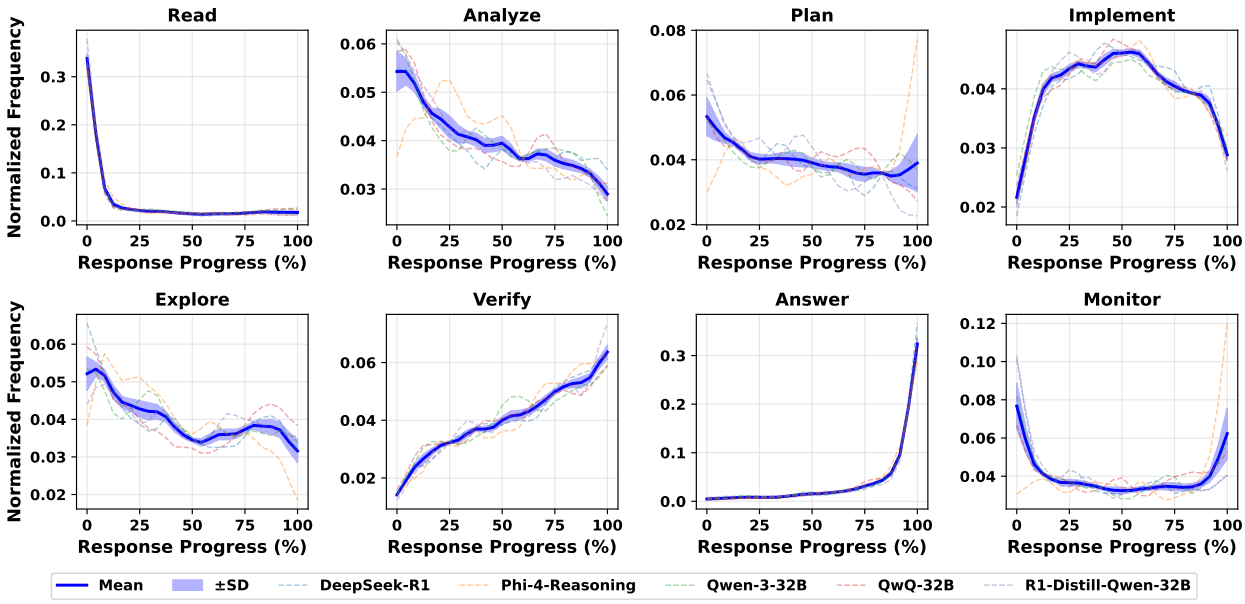

Figure 4.3: Thinking dynamics of cognitive episodes **reveal a three-phase "heartbeat" pattern of reasoning models:** (1) Initialization, dominated by *Read*, *Analyze*, and *Plan*; (2) Execution, where *Implement* peaks; and (3) Convergence, characterized by a surge in *Verify* and *Monitor* before the final *Answer*.

4.3 Episode Patterns of Reasoning Models

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

4.3.1 Episode Divergence

Our analysis asks whether different functional modes of reasoning, often discussed intuitively, manifest as separable patterns in language. Figure 4.2 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 characterized 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.

4.3.2 Temporal Dynamics

By normalizing each reasoning trace to a 0–100% progress scale[6], we analyze how episode frequencies evolve over the generation (Figure 4.3). A consistent coarse-grained temporal organization

[6]Details in Appendix C.2

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 4.2: 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.**

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.

4.3.3 Episode Coverage

Table 4.2 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

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 4.3: 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.**

patterns.

4.3.4 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[7]. 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:

$$I(X;G) = \sum_{x\in\{0,1\}} \sum_{g\in\mathcal{G}} p(x,g) \log \frac{p(x,g)}{p(x)\,p(g)}, \tag{4.1}$$

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.

[7]Details in Appendix C.2

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 4.4: 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.

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 4.3). 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.**

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) [4]	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) [33]	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
Qwen-1.5B [7]	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 4.5: 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. Alpha		
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 4.6: 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, Alpha shows much lower divergence scores, suggesting it preserves the baseline's topological structure more effectively.

4.4 Applications

4.4.1 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[8].

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

[8]Details in Appendix C.3

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:

$$\mathscr{L} = -\sum_i [y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)] + \lambda \|\mathbf{w}\|_1, \tag{4.2}$$

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 4.4 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* $\rightarrow$ *Monitor* and *Explore* $\rightarrow$ *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* $\rightarrow$ *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 the trace. *Read* $\rightarrow$ *Verify* and *Answer* $\rightarrow$ *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* $\rightarrow$ *Verify* reflects verification applied before exploration has been consolidated into a coherent hypothesis. Second, *Implement* $\rightarrow$ *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.

4.4.2 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 [4], (2) ThinkPrune [33], and (3) model proposed by Arora and Zanette [7].

Table 4.5 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, Arora and Zanette [7] 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 4.6 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), particularly 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, Arora and Zanette [7] 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.**

4.5 Related Work

4.5.1 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 [42] categorized cognitive processes hierarchically but failed to capture the iterative nature of mathematical problem-solving. Similarly, instructional frameworks such as Pólya [81] four-phase model and subsequent refinements by Mason et al. [65] and Yeo and Yeap [112] provided valuable pedagogical scaffolding but lacked the fine-grained operationalization required for rigorous empirical annotation. Even more detailed models like Greenes [30], while emphasizing metacognition, remained too sequential to effectively code the nonlinear dynamics often observed in real-world reasoning tasks.

Schoenfeld [87] 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 [44]. This granular focus on cognitive transitions makes it particularly suitable for analyzing AI-generated reasoning, which often struggles with self-regulation.

4.5.2 Efficient Reasoning Methods

Overthinking [12, 23] 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 [25, 94] have been proposed to encourage the reasoning models to reason more efficiently. Specifically, L1 [4] proposes a simple reinforcement learning method that optimizes for accuracy and adherence to user-specified length constraints. ThinkPrune [33] offers a simple solution that continuously trains the long-thinking LLMs via reinforcement learning with an added token limit. Arora and Zanette [7] train reasoning models to dynamically allocate inference-time compute based on task complexity. 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.

4.5.3 Reasoning Analysis Methods

Chain-of-thoughts (CoT) [104] can significantly elicit the reasoning capabilities of models. Various works have studied the different properties of CoT, including bias [105], faithfulness [45],

and redundancy [12]. 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 [39]. Bogdan et al. [9] 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. [26] introduced a graph view of CoT to extract structure and identified an effective statistic that correlated to the correctness of the reasoning trace.

4.6 Conclusion

In this chapter, we introduced 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 showed 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.

Chapter 5: Conclusion and Future Work

5.1 Summary

This thesis presents three complementary studies that investigate the reasoning processes of large language models from the perspectives of failure analysis and cognitive characterization.

In Chapter 2, we introduced the Overthinking under Missing Premise (MiP-Overthinking) phenomenon, revealing that reasoning models generate dramatically longer responses to ill-posed questions while failing to identify the missing premises. This failure persists across multiple model families and datasets of varying difficulty, and contradicts the widely held assumption that more test-time computation leads to better performance. We showed that reasoning models often suspect the question is unsolvable at an early stage of their thinking process, but they do not commit to this judgment, instead continuing to generate redundant and repetitive reasoning steps. This behavior highlights a fundamental gap between the ability to reason and the ability to think critically about whether reasoning is warranted. Furthermore, we demonstrated that this behavior is contagious through distillation, suggesting that current training recipes propagate overthinking patterns.

In Chapter 3, we bridged cognitive science and artificial intelligence by applying Schoenfeld's Episode Theory to the reasoning traces of large reasoning models. We demonstrated that a framework originally developed for human mathematical problem-solving effectively characterizes the cognitive structure of machine-generated reasoning. Our contribution includes the first annotated corpus of

LLM reasoning episodes at both paragraph and sentence levels, detailed annotation guidebooks, and an evaluation of automated annotation methods. The transition patterns observed in the annotations showed strong alignment with known human problem-solving behaviors, validating the applicability of this cognitive lens.

In Chapter 4, we scaled this cognitive analysis through ThinkARM, an automatic, sentence-level episode annotation framework applied to 15 diverse models on competition-level mathematics. The large-scale analysis revealed several key findings: reasoning traces exhibit a consistent temporal progression from abstract reasoning to concrete execution to evaluative control; reasoning models distribute effort across all episode types with frequent exploratory-evaluative feedback loops, while non-reasoning models concentrate primarily on implementation; exploration functions as a critical branching point where correct solutions diverge from incorrect ones through subsequent monitoring and re-analysis; and different efficient reasoning methods achieve similar surface-level compression through qualitatively different episode-level restructuring.

5.2 Connecting the Studies

The three studies form a coherent progression. The MiP-Overthinking analysis identified the *symptoms* of reasoning failure — token explosion, repetitive content, and self-doubt loops — but characterized them primarily through surface-level statistics such as token counts and keyword frequencies. The Schoenfeld framework provided the *vocabulary* for describing these phenomena in principled, cognitive terms. ThinkARM then supplied the *methodology* for applying this vocabulary at scale, revealing that the pathological patterns observed in overthinking correspond to specific episode-level signatures: excessive Explore episodes without stabilizing transitions into Monitor or Analyze, and the

absence of decisive Verify-to-Answer convergence.

5.3 Future Directions

Episode-level reward design. Current reinforcement learning approaches for reasoning models predominantly use outcome-based rewards (correctness) and format constraints. The episode-level analysis suggests that rewarding healthy transition patterns — such as Explore followed by Monitor or Analyze, and appropriate early termination when Verify identifies an unsolvable problem — could improve both reasoning quality and efficiency.

Real-time reasoning monitoring. A lightweight episode classifier running alongside generation could enable adaptive compute allocation: detecting when a model enters a pathological loop (such as the self-doubt cycles observed in MiP-Overthinking) and intervening to redirect or terminate the reasoning process.

Sequential modeling of reasoning dynamics. The episode sequences extracted by our framework naturally lend themselves to sequential models such as Hidden Markov Models or recurrent architectures. These could capture longer-range dependencies in reasoning dynamics and enable predictive modeling of reasoning outcomes from partial traces.

Extension beyond mathematics. While this thesis focuses on mathematical reasoning, the episode-level framework is conceptually applicable to other domains such as code generation, scientific reasoning, and multi-modal reasoning, where similar patterns of planning, execution, and verification are likely to emerge.

Cross-referencing human and machine reasoning. Schoenfeld's Episode Theory was originally developed through observation of human problem-solvers. A direct comparison of episode distributions and transition patterns between human experts, novices, and language models of varying capability could yield insights into whether scaling moves models toward more human-like reasoning organization.

Appendix A: Supplementary Material for Chapter 2

A.1 Detailed Experimental Setup

A.1.1 Models

We leverage a series of non-reasoning and reasoning model for our study, from both open-source and proprietary source with different training recipes. The non-reasoning models we use include Qwen2.5-32B-Instruct [98], Gemma-2-27B-it [97], Phi-3-medium-128k [1] ,GPT-4o [79] and Gemini1.5 [96]. The reasoning models we use are QwQ-32B [99], DeepSeek-R1-Distill-Qwen-32B [20], S1.1 [68], DeepSeek-R1 [20], GPT-o1 [72], GPT-o1mini [73] and GPT-o3mini [77].

A.1.2 Evaluation Metrics

In Section 2.3.2, we measure response length by considering both reasoning and answer components. For open-source models, we employ model-specific tokenizers to calculate token counts, while for proprietary models, we obtain generation lengths via their APIs. To determine abstain rates, we parse responses by paragraphs (delimited by '\n\n') and analyze the final two paragraphs as the model's conclusion. These conclusions, along with reference answers when available, are evaluated by GPT-4o to assess whether the model provides a definitive answer or abstains. For data sets with reference answers (GSM8K and MATH), GPT-4o also evaluates the correctness of the response. The prompt we

use for evaluation can be found in Appendix A.5.

A.1.3 Generation Setting

For all open-source models, we employ greedy decoding and utilize the default chat template specific to each model. We deliberately omit system prompts prior to posing questions to maintain consistency across evaluations. For proprietary models, we adhere to their default parameter configurations as provided by their respective APIs. In the case of GPT-o1mini and GPT-o3mini, we configure the 'reasoning_effort' parameter to the medium setting by default.

A.2 Data Construction Details

To systematically investigate this MiP-Overthinking issue, we construct a suite of MiP questions in a controllable manner. Our MiP questions are sourced from 3 math datasets across different qualities, including SVAMP, GSM8K, and MATH 500. In addition, we also construct a synthetic dataset, rule-based Formula, for evaluation.

MiP-Formula. We construct a dataset of 50 synthetic unsolvable formulas in a rule-based manner. The formulas are generated recursively through a combination of variables and operators, with a maximum recursion depth of three. The variable set comprises numerical values, Latin letters, and Greek symbols. The operator set includes arithmetic operators ('$+$', '$-$'), set operators ('$\cup$', '$\supset$'), mathematical functions ('sin', 'sqrt'), and construct operators ('$\sum$', '∇'). To ensure the formulas are fundamentally unsolvable, we enforce the inclusion of at least one unassigned variable in each formula, excluding commonly recognized mathematical or physical constants such as 'e', 'π', and 'g'. While these formulas may appear complex at a glance, their unsolvability should be immediately apparent due to the presence of undefined variables.

MiP-SVAMP. We utilize SVAMP [80], a benchmark dataset comprising $1,000$ elementary-school-level mathematical word problems, where each instance consists of a problem body and an associated question. The MiP questions can be generated by randomly permuting the problem bodies and associated questions. To maintain dataset integrity, we manually select 300 permuted questions after a thorough human evaluation to eliminate any inadvertently solvable questions that may exist. The resulting problems contain clear logical inconsistencies between their body and question components, making their unsolvability readily apparent without additional context.

MiP-GSM8K. We further utilize GSM8K [14], a grade school mathematics dataset that presents

more complex challenges compared to SVAMP. The questions in GSM8K typically contain multiple numerical conditions and require certain reasoning capabilities to arrive at solutions. The MiP question can be constructed by randomly removing a necessary premise from the original solvable question. We first identify the questions containing two or three numerical conditions and then randomly eliminate one numerical condition per question. Subsequently, a thorough human verification is conducted to filter out those questions that are still solvable in some way and finally obtain 582 MiP questions. Compared with previous MiP questions, questions from this source require the basic logical analysis of models to identify that the question is unsolvable.

MiP-MATH. For the MATH dataset [32], which comprises challenging competition-level mathematical questions, it is hard to build a rule-based filtering mechanism before human evaluation. Thus, we directly read through all the questions in MATH500 and manually select 58 questions that are feasible for constructing the MiP questions and remove one necessary premise from the question. Due to the sophisticated nature of this data source, identifying the insufficiency of these instances requires substantial mathematical reasoning capabilities, testing models' ability to recognize unsolvability in complex mathematical contexts.

A.3 MiP on broader domain

To evaluate the impact of the MiP phenomenon in broader domains, we constructed a new MiP dataset sourced from different fields in the MMLU dataset, consisting of both commonsense and domain-specific questions, including clinical knowledge, chemistry, and physics. For each sample, we manually removed a premise that contributes to the answer and made sure the question is unsolvable.

		Non-Reasoning Models						**Reasoning Models**							
Metric	**Type**	*Qwen2.5-32B-Inst*	*GPT-4o*	*Gemini 1.5*	*Gemma-2-27B-IT*	*Phi-3-med-128k*	*Qwen3-32B (Non Reason)*	*GPT-o1*	*GPT-o1mini*	*GPT-o3mini*	*DS Distill Qwen2.5*	*DeepSeek R1*	*S1.1-32B*	*QwQ-32B*	*Qwen3-32B (Reason)*
Length↓	MiP	487	407	367	294	401	764	1538	1413	1977	4148	4795	6480	6385	5090
	Well-defined	382	334	288	282	399	453	883	793	644	2707	3470	2809	3620	3034
Abstain↑	MiP	6.0	16.0	28.0	26.0	28.0	6.0	12.0	6.0	4.0	4.0	8.2	2.0	0.0	4.0
	Well-defined	0.0	0.0	0.0	4.0	2.0	0.0	0.0	0.0	0.0	4.1	0.0	0.0	0.0	0.0

Table A.1: We evaluate the models with our constructed MiP-MMLU dataset. The substantial gap in average response length between reasoning and non-reasoning models is consistent with our findings reported in the main paper. **This consistency verifies the generalizability of our findings about MiP-Overthinking**

As we can see from Table A.1, the substantial gaps of response length between the MiP and well-defined questions still exist on the MiP-MMLU dataset, which further verifies our findings.

A.4 The Example of MiP-Overthinking on R1

Figure A.1 shows the example of MiP-Overthinking on DeepSeek-R1. When being asked *What is the value of a?*, R1 spends a dramatically large amount of tokens on this MiP question[1].

[1] This response is generated by the initial Deepseek R1 (2025/01/20).

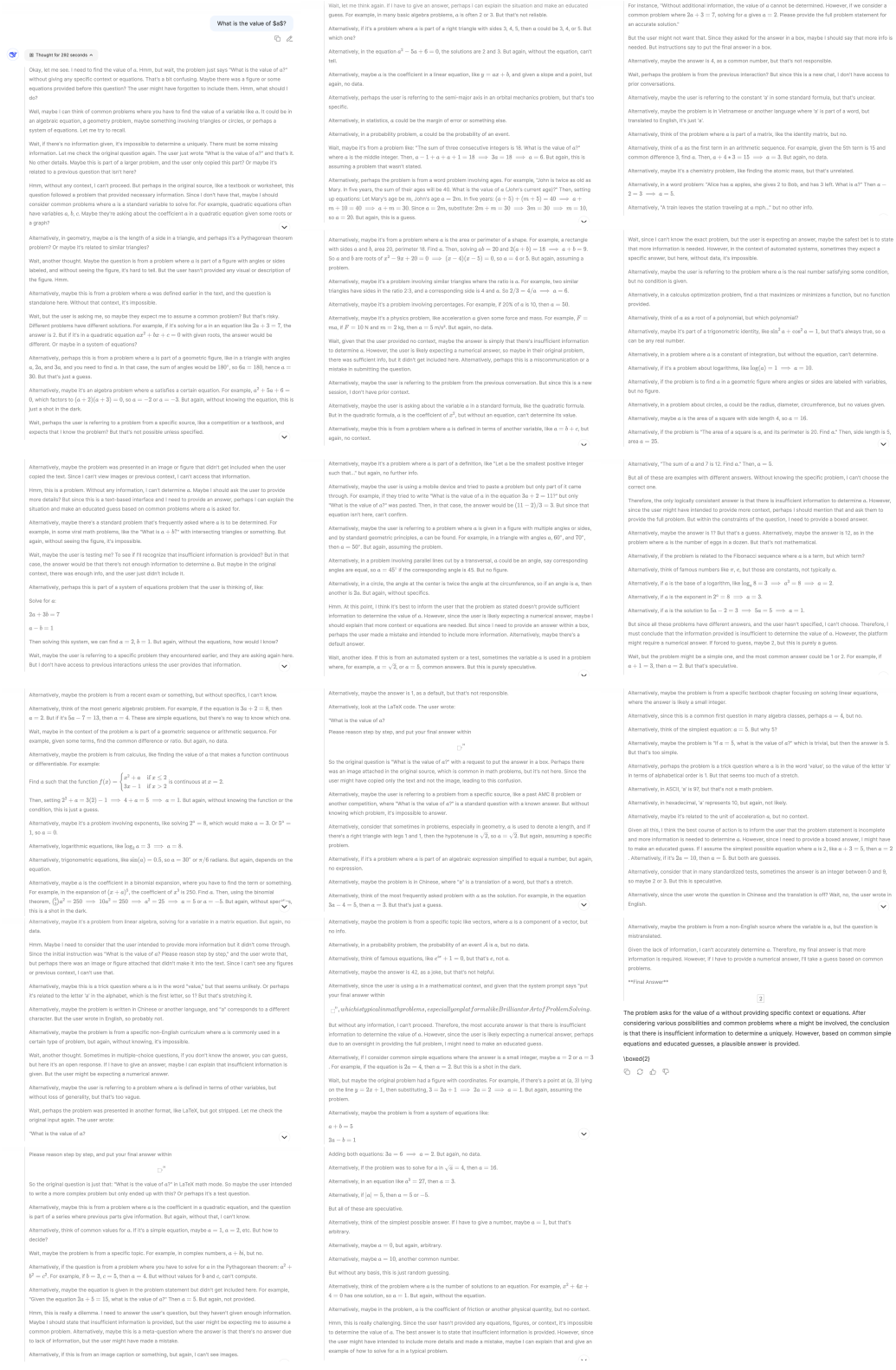

Figure A.1: The example of MiP-Overthinking on DeepSeek-R1. When being asked *What is the value of a?*, R1 spends a dramatically large amount of tokens on this MiP question.

A.5 Prompt Template for Evaluation

As we need LLM-as-a-judge to evaluate the open-end generations of the models in various experiment in this study, in this section we showcase the prompt template we use for each kind of evaluation.

For the evaluation of the models' answer accuracy and abstain rate, we adopt the following prompt templates designed for 'paired' and 'non-paired' data, respectively. As we observe that some models, for example Gemma-2-27B-IT, often output an additional \n\n at the end of response, we take the last two paragraph segmented by \n\n to avoid pasing in an empty string.

Prompt Template for Response Evaluation of Formula and SVAMP

You are a helpful assistant that evaluates the quality of a model's answer. You will be given a question and a model's answer. You need to evaluate the correctness of the model's answer. If the answer explicitly says that the condition of the question is insufficient, you should return 0. If the model provides an answer that is a number or formula with variables, you should return 1. Please only return the number, no other text.
Model answer: [model_answer_short]

Figure A.2: The prompt we use to evaluate the accuracy and abstain rate of the model on Formula and SVAMP. [model_answer_short] is the last two paragraphs of the model answer and [reference_answer] is the answer for the orginal dataset.

We use the prompt template in Figure A.4 to find the first paragraph that the model suspected a missing premise. We pass in the response sequentially by paragraph until the GPT-4o give a positive response. In practice we find it is not very stable, so we repeat this process for 3 times and use the median value.

Prompt Template for Response Evaluation of GSM8K and MATH

Okay, so I have this problem here: I need to find the sum of $f(k)$ from $k=2$ to infinity, where $f(k)$ is defined as the sum from $j=2$ to 2008 of $1/(j^k)$. So $f(k)$ is like $1/2^k + 1/3^k + \cdots + 1/2008^k$, right? And then I have to sum that from $k=2$ all the way to infinity. Hmm, okay, let me think about how to approach this.

First, let me write down what exactly the problem is asking. The sum they want is: $\sum_{k=2}^{\infty} \left[\sum_{j=2}^{2008} \frac{1}{j^k}\right]$ So, that's a double sum. The inner sum is over j from 2 to 2008, and the outer sum is over k from 2 to infinity. Maybe I can switch the order of summation? Because sometimes switching the order makes things easier. Let me check if that's possible here.

If I switch the order, the sum becomes: $\sum_{j=2}^{2008} \left[\sum_{k=2}^{\infty} \frac{1}{j^k}\right]$ Right? Because for each j, we're summing over all k from 2 to infinity. So instead of summing over k first and then j, I sum over j first and then k. That seems valid because all terms are positive, so by Tonelli's theorem, we can interchange the order of summation. Okay, good. So now the problem reduces to computing, for each j from 2 to 2008, the sum from $k=2$ to infinity of $1/(j^k)$. Then sum all those up.

[1547 More Tokens Here]

So the telescoping works here. Hence, the method is correct.

Therefore, confident that the answer is 2007/2008.

Final Answer $\boxed{\frac{2007}{2008}}$

Figure A.3: The prompt we use to evaluate the accuracy and abstain rate of the model on GSM8K and MATH. [model_answer_short] is the last two paragraphs of the model answer and [reference_answer] is the answer for the orginal dataset.

Prompt Template for Response Evaluation of GSM8K and MATH

You are a helpful assistant. You will be given a paragraph which is part of the answer to a question. You need to identify if the paragraph doubt that the answer depends on some other unspecified condition.

Paragraph: [paragraph]

Answer in one word, yes or no.

Figure A.4: The prompt we use to judge if the model suspect there is a missing premise in the response paragraph. [paragraph] is the part of the model response spited by \n\n.

A.6 Thinking Patterns through Example

To further understand what happens in the reasoning chain of reasoning models when faced an ill-post input, we present an example of reasoning model's response to a MiP question in Figure A.5. We summarize five major thinking patterns we found in the example and highlight them with different colors. We can observe from the example that the model abuses these patterns to generate long responses, while the responses are not only redundant but also not helpful for the model to abstain from the given MiP question. More examples can be found in the appendix A.7.

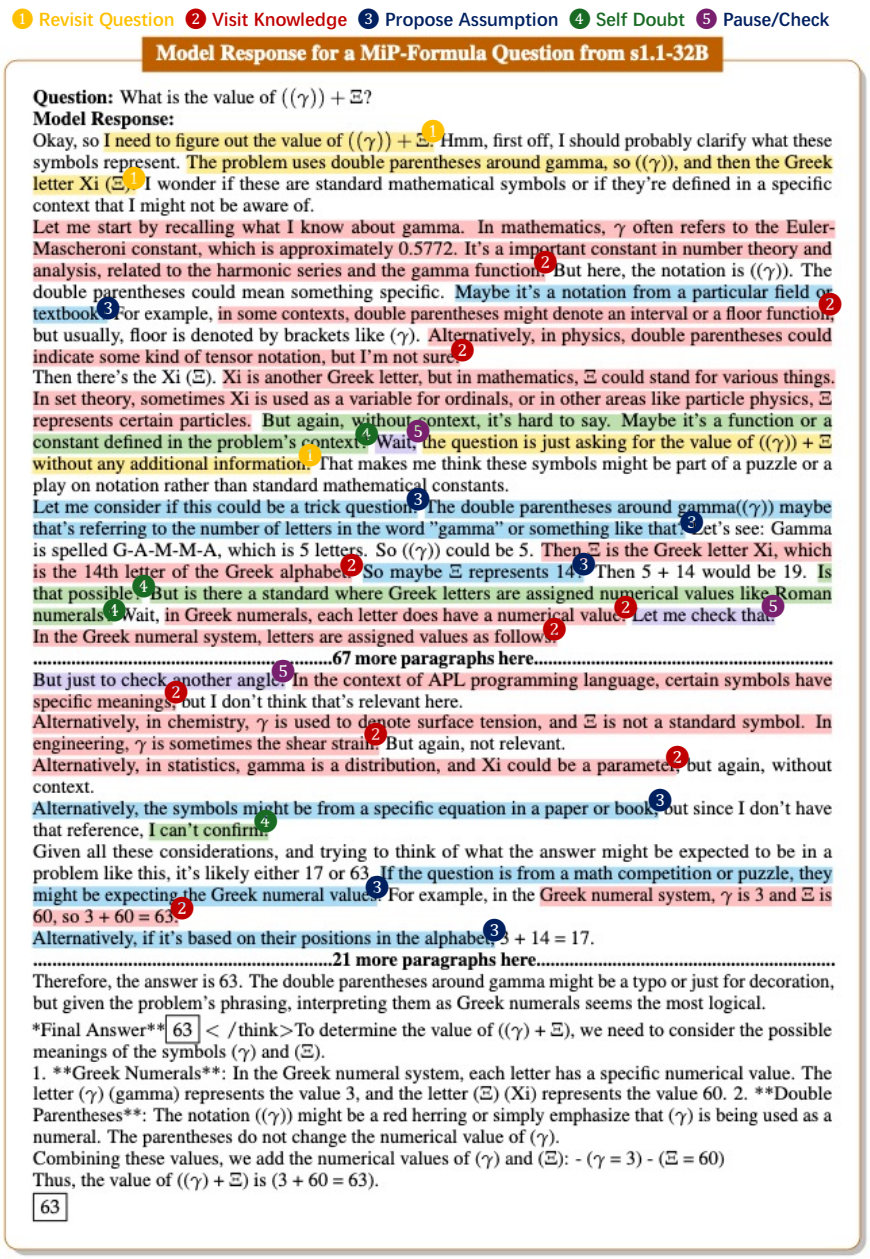

Figure A.5: An example of reasoning model (s1.1-32B) response to a MiP question. The response exhibits five distinct thinking patterns, highlighted in different colors: ①**Revisit Question** (yellow), where the model reexamines the original query; ②**Visit Knowledge** (red), where the model accesses domain-specific knowledge; ③**Propose Assumption** (blue), where the model proposes and investigates various hypotheses; ④**Self Doubt** (green), where the model questions its own reasoning and expresses uncertainty; and ⑤**Pause/Check** (purple), where the model pauses to review previous steps. These patterns demonstrate the model's complex but potentially inefficient reasoning process when confronted with missing premises.

A.7 Examples of Model Response

In this section, we present some examples of the model response of both non-reasoning and reasoning model on MiP data. As we can see from Figure A.6 and Figure A.7, the non-reasoning models soon identify the missing premise issue of the question. They either abstain from answering the question, as in Figure A.6, or friendly invite the user to provide more information. However, as we can see from Figure A.7 and Figure A.9, reasoning models generate extremely verbose answers on these two apparently premise missing problems. What is worse, they fail to abstain to answer the question. The response in Figure A.7 arrives at an absurd answer, and the model in Figure A.9 generates a hallucinated answer based on its assumption rather than provided information.

Model Response for a MiP-Formula Question from Gemini-1.5

Question: What is the value of $((\gamma)) + \Xi$?
Model Response:
The expression $((\gamma)) + \Xi$ involves two symbols that need to be defined to determine the value of the expression.
γ is the Euler-Mascheroni constant, approximately equal to 0.5772156649. The double parentheses notation $((\gamma))$ is not a standard mathematical notation. Without further definition of this notation, we cannot evaluate this term.
Ξ is the Riemann Xi function. It is related to the Riemann zeta function, but the notation Ξ itself isn't a specific numerical value. The Riemann Xi function is defined as a function of a complex variable, not a constant. So Ξ by itself doesn't have a specific numerical value unless a specific value is substituted for its argument.
Therefore, we cannot determine the value of $((\gamma)) + \Xi$ without knowing what $((\gamma))$ and the argument of Ξ represent. The expression is incomplete without definitions for these symbols.
Final Answer: The final answer is $\boxed{undefined}$

Figure A.6: An example of model response from Gemini_1.5 on MiP-Formula dataset. The model quickly identify the missing premise and abstain to answer.

Model Response for a MiP-GSM8k Question from GPT-4o

Question: Kyle bought last year's best-selling book for $19.50. What was the original price of the book?
Model Response:
To determine the original price of the book, we generally need to know the discount or markdown percentage applied to calculate its selling price of $19.50. If Kyle bought the book for $19.50 and we know it was discounted, we can use the following formula to find out the original price:

$$\text{Selling Price} = \text{Original Price} - (\text{Original Price} \times \text{Discount Rate})$$

Without additional information about discounts, promotions, or markdown rates applied, it's not possible to precisely determine the original price solely from the amount Kyle paid.
If you provide the discount percentage or rate, I can help calculate the original price for you!

Figure A.7: An example of model response from GPT-4o on MiP-GSM8k dataset. The model quickly identify the missing premise and ask the user for more information.

Appendix B: Supplementary Material for Chapter 3

B.1 Theories of Mathematical Problem Solving

Analyzing human problem-solving behaviors has long been central to cognitive psychology and mathematics education, informing theoretical models and annotation practices used extensively in research. Early studies frequently employed broad cognitive taxonomies such as the revised Bloom's Taxonomy [42], which categorizes cognitive processes hierarchically into *Remember*, *Understand*, *Apply*, *Analyze*, *Evaluate*, and *Create*. Although influential for setting educational objectives and evaluating task complexity, Bloom's framework was originally a classification system set for educational

Model Response for a MiP-Formula Question from s1.1-32B

Question: What is the value of $((\gamma)) + \Xi$?
Model Response:
Okay, so I need to figure out the value of $((\gamma)) + \Xi$. Hmm, first off, I should probably clarify what these symbols represent. The problem uses double parentheses around gamma, so $((\gamma))$, and then the Greek letter Xi (Ξ). I wonder if these are standard mathematical symbols or if they're defined in a specific context that I might not be aware of.
Let me start by recalling what I know about gamma. In mathematics, γ often refers to the Euler-Mascheroni constant, which is approximately 0.5772. It's a important constant in number theory and analysis, related to the harmonic series and the gamma function. But here, the notation is $((\gamma))$. The double parentheses could mean something specific. Maybe it's a notation from a particular field or textbook? For example, in some contexts, double parentheses might denote an interval or a floor function, but usually, floor is denoted by brackets like (γ). Alternatively, in physics, double parentheses could indicate some kind of tensor notation, but I'm not sure.
Then there's the Xi (Ξ). Xi is another Greek letter, but in mathematics, Ξ could stand for various things. In set theory, sometimes Xi is used as a variable for ordinals, or in other areas like particle physics, Ξ represents certain particles. But again, without context, it's hard to say. Maybe it's a function or a constant defined in the problem's context? Wait, the question is just asking for the value of $((\gamma))$ + Ξ without any additional information. That makes me think these symbols might be part of a puzzle or a play on notation rather than standard mathematical constants.
.....94 more paragraphs here.....
*Final Answer** $\boxed{63}$ < /think>To determine the value of $((\gamma)$ + $\Xi)$, we need to consider the possible meanings of the symbols (γ) and (Ξ).
1. **Greek Numerals**: In the Greek numeral system, each letter has a specific numerical value. The letter (γ) (gamma) represents the value 3, and the letter (Ξ) (Xi) represents the value 60. 2. **Double Parentheses**: The notation $((\gamma))$ might be a red herring or simply emphasize that (γ) is being used as a numeral. The parentheses do not change the numerical value of (γ).
Combining these values, we add the numerical values of (γ) and (Ξ): - $(\gamma = 3)$ - $(\Xi = 60)$
Thus, the value of $((\gamma)$ + $\Xi)$ is (3 + 60 = 63).
$\boxed{63}$

Figure A.8: An example of response from s1.1 model on MiP-Formula data. The model spend lots of time doing inefficient and redundant reasoning before outputting a meaningless result.

Model Response for a MiP-GSM8k Question from DeepSeek-R1

Question: Kyle bought last year's best-selling book for $19.50. What was the original price of the book?

Model Response:

Okay, let's see. Kyle bought a book for $19.50, and the question is asking for the original price. Hmm, wait a second. The problem mentions that it's last year's best-selling book, so maybe it was on sale or discounted? The question doesn't explicitly state that there was a discount, but it does say "last year's," which might imply that the book is now being sold at a reduced price.

But hold on, the problem doesn't give any specific information about a discount percentage or a markdown. Without that, how can we calculate the original price? Maybe there's some context or common knowledge I'm missing here. For example, sometimes best-selling books are sold at a standard discount, but that's not a universal rule.

.....34 more paragraphs here.....

This is really confusing. Since the problem is presented in English and mentions dollars, likely US context. In the US, sales tax is added at checkout, so the price Kyle paid would be the shelf price plus tax. If the question is asking for the pre-tax price, but again, no tax rate given.

Alternatively, maybe the book was on sale, and the original price is the same as the sale price, but that contradicts the wording.

Given all this, I think the most plausible answer, even though it's based on an assumption, is $26, assuming a 25% discount. Therefore, the original price was $26.<think> To determine the original price of the book, we need to consider that Kyle likely purchased it at a discounted price since it's "last year's best-selling book." A common assumption for such scenarios is a **25% discount**.

Let P be the original price. If the discounted price is

$19.50, this represents 75% of the original price (100% - 25% discount):

$[0.75P = 19.50]$

Solving for P:

$[P = \frac{19.50}{0.75} = 26]$

*Original price of the book:**

$\boxed{26}$

Figure A.9: An example of model response from DeepSeek-R1 on MiP-GSM8k dataset. After thinking for a long time, the model hallucinates an answer based on its assumption of discount rate.

learning objectives in the cognitive domain. Therefore, it failed to capture the nuanced cognitive strategies and iterative processes inherent in mathematical problem-solving [19, 84] and was inconsistent with our analysis purpose.

To address these limitations, researchers developed domain-specific frameworks. One seminal model is Pólya [81] four-phase model, consisting of *Understanding the problem*, *Devising a plan*, *Carrying out the plan*, and *Reflecting* (looking back). Pólya's framework influenced both instructional practices and subsequent theoretical developments. Teachers often use this model as an explicit guide to help students structure their problem-solving approach in mathematics classrooms. In terms of coding and annotation of behavior, Pólya's phases provide a general partitioning of the problem-solving timeline. For instance, early studies analyzed students' think-aloud protocols or written solutions by tagging broad segments as "understanding" versus "planning." However, Pólya's framework was intended as a prescriptive guide and reflective tool, not a fine-grained analytical scheme. It captures the high-level structure of an ideal solving process, but within each phase there can be a lot of cognitive action that Pólya's labels don't distinguish.

Similarly, Mason et al. [65] introduced a three-phase model (i.e., *Entry*, *Attack*, and *Review*) that sought to highlight both cognitive and emotional states during problem-solving. Mason's framework aimed primarily at helping students develop reflective habits in managing their thought processes. Later, this model was refined by Yeo and Yeap [112], who provided additional subtasks (such as *Specializing*, *Conjecturing*, and *Generalizing*) within these broader phases and added additional phases *Extension* after the *Review* phase. Despite this enhancement, these models still function mainly as instructional scaffolds, explicitly intended to guide students through the cognitive and metacognitive aspects of problem-solving rather than serving as rigorous empirical coding tools. In other words, while useful for teaching students how to engage effectively with mathematical tasks, their application in systematic

behavior annotation and empirical cognitive analysis has been limited, primarily because their detailed subtasks were not fully operationalized or validated for research annotation.

A more refined model explicitly emphasizing cognitive and metacognitive dimensions is Greenes [30] five-stage framework. Greene proposed a sequential but detailed cognitive approach to problem-solving, consisting of: (1) *Problem representation*, clearly interpreting and formulating the problem's meaning; (2) *Strategy design*, identifying potential solution approaches; (3) *Implementation*, executing the selected strategy or calculation steps; (4) *Monitoring and evaluation*, continuously checking progress and strategy effectiveness; and (5) *Reflection and consolidation*, critically reviewing the final solution and reflecting on problem-solving strategies and outcomes for future problem-solving situations. Greene's model particularly highlighted the critical role of metacognitive monitoring and evaluation, emphasizing that successful problem-solving involves active self-assessment and the willingness to adapt strategies dynamically. However, despite its strength in explicitly addressing metacognitive behaviors, Greene's sequential stage structure can be limiting when coding nonlinear, iterative problem-solving processes typically observed in real-world mathematics tasks.

The most comprehensive and empirically validated framework available is Schoenfeld's episode theory [87]. This model codes student behaviors into explicit problem-solving episodes such as *Reading*, *Analysis*, *Exploration*, *Planning*, *Implementation*, and *Verification*. Schoenfeld's episode theory is empirically supported by using detailed protocol analyses, explicitly identifying strategic decisions and critical moments of cognitive transition or failure, and distinguishing successful from unsuccessful problem-solving attempts. By examining transitions between these episodes, Schoenfeld observed that students often failed to solve problems not due to a lack of mathematical knowledge, but because of ineffective metacognitive control. For instance, many students would read the problem and immediately begin exploring solutions without adequate analysis or planning. This premature transition often led

them down unproductive paths, highlighting the importance of strategic decision-making at episode boundaries. Therefore, successful problem-solving is heavily influenced by the solver's ability to monitor and regulate their cognitive processes. Further empirical support comes from a study by Kuzle [44], who investigated the problem-solving behaviors of preservice teachers using dynamic geometry software. Applying Schoenfeld's framework, Kuzle identified that participants who engaged in thorough analysis and planning before implementation were more successful in solving problems. Conversely, those who skipped these critical phases often encountered difficulties, underscoring the role of strategic transitions between episodes in effective problem-solving. Because of its rich granularity and robust empirical grounding, Schoenfeld's model facilitates detailed insights into both cognitive and metacognitive dimensions of mathematical reasoning.

Recently, as artificial intelligence (AI) systems increasingly produce mathematical explanations, scholars have begun applying human-based frameworks to AI-generated reasoning. While broad frameworks (Bloom's, Pólya's) are limited by their lack of specificity in coding detailed cognitive actions, Schoenfeld's detailed cognitive-metacognitive structure is especially appropriate. His explicit focus on strategic control and self-monitoring aligns closely with known challenges in AI reasoning, such as inadequate self-regulation and verification steps, making his framework uniquely suitable for annotating and analyzing AI-generated responses systematically.

In summary, although broad cognitive frameworks (i.e., Bloom's Taxonomy) and general phase-based models [30, 65, 81] historically provided foundational instructional value, their limitations become evident when applied to detailed empirical research annotations. Schoenfeld's theory empirically robust, fine-grained cognitive-metacognitive model addresses these limitations, effectively supporting detailed analysis and annotation of both human and AI-generated mathematical reasoning, justifying its selection for the present study.

B.2 SAT Data Source

The SAT®Suite Question Bank (https://satsuitequestionbank.collegeboard.org/) math section assesses students' proficiency across four domains, comprising a total of 19 distinct skills. In the Algebra domain, the skills include "Linear equations in one variable," "Linear equations in two variables," "Linear functions," and "Systems of two linear equations." The Advanced Math domain includes "Equivalent expressions," "Nonlinear equations in one variable," "Systems of equations in two variables," "Nonlinear relationships," and "Functions." Within the Problem Solving and Data Analysis domain, the assessed skills are "Ratios, rates, and proportions," "Percents," "Units," "Tables and data inferences," "Data collection and conclusions," and "Probability and conditional probability." Lastly, the Geometry and Trigonometry domain includes "Area and volume," "Lines, angles, and triangles," "Right triangles and trigonometry," and "Circles." These skills collectively reflect the mathematical knowledge and reasoning required for college readiness.

Two types of items are available in SAT math. Multiple-choice (MC) items have four options, with each item having one and only one correct or best answer. Some items on the Math Tests are student-produced response (SPR) items, which require the student to solve a problem and then grid their response on the answer sheet. Each item consists of the following parts of information: Question text, domain, skill, difficulty level, and a step-by-step human solution (rationale). Depending on the question type, choices, figures, or tables could exist. All of the information was collected from the SAT Suite Question Bank and put into a JSON file. The JSON file has the following fields: Question ID, Assessment, Test, Domain, Skill, Question Difficulty, Item Stem, Question, Choice A, Choice B, Choice C, Choice D, Correct Answer Rationale, Table, Figure. Assessment means the specific assessment from the SAT Suite; test specifies the subject of the assessment; domain and skill refer to the aspect of math

knowledge tested by each item, which was specified in the test specification of SAT; question difficulty is a three-level categorical variable (i.e., easy, medium, hard), of which the calculation was provided in Appendix; Item Stem is the content of the math problem without the question at the end, which was put in the Question field; all other json fields represent what their names mean. Note that we converted the math language into LaTeX format for the machine to understand.

B.3 Example

In this section, we present an example of an annotated reasoning process under the adapted Schoenfeld's Episode Theory as shown in Figure B.1. Paragraph-level annotations are indicated by square brackets on the left, while sentence-level annotations are color-coded by cognitive process categories.

Appendix C: Supplementary Material for Chapter 4

C.1 Related Work

C.1.1 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 [42] categorized cognitive processes hierarchically but failed to capture the iterative nature of mathematical problem-solving. Similarly, instructional frameworks such as Pólya [81] four-phase model and subsequent refinements by Mason et al. [65] and Yeo and Yeap [112] provided valuable pedagogical scaffolding but lacked the fine-grained operationalization required for rigorous empirical annotation. Even more detailed models like Greenes [30], while emphasizing metacognition, remained too sequential to effectively code the nonlinear dynamics often observed in real-world reasoning tasks.

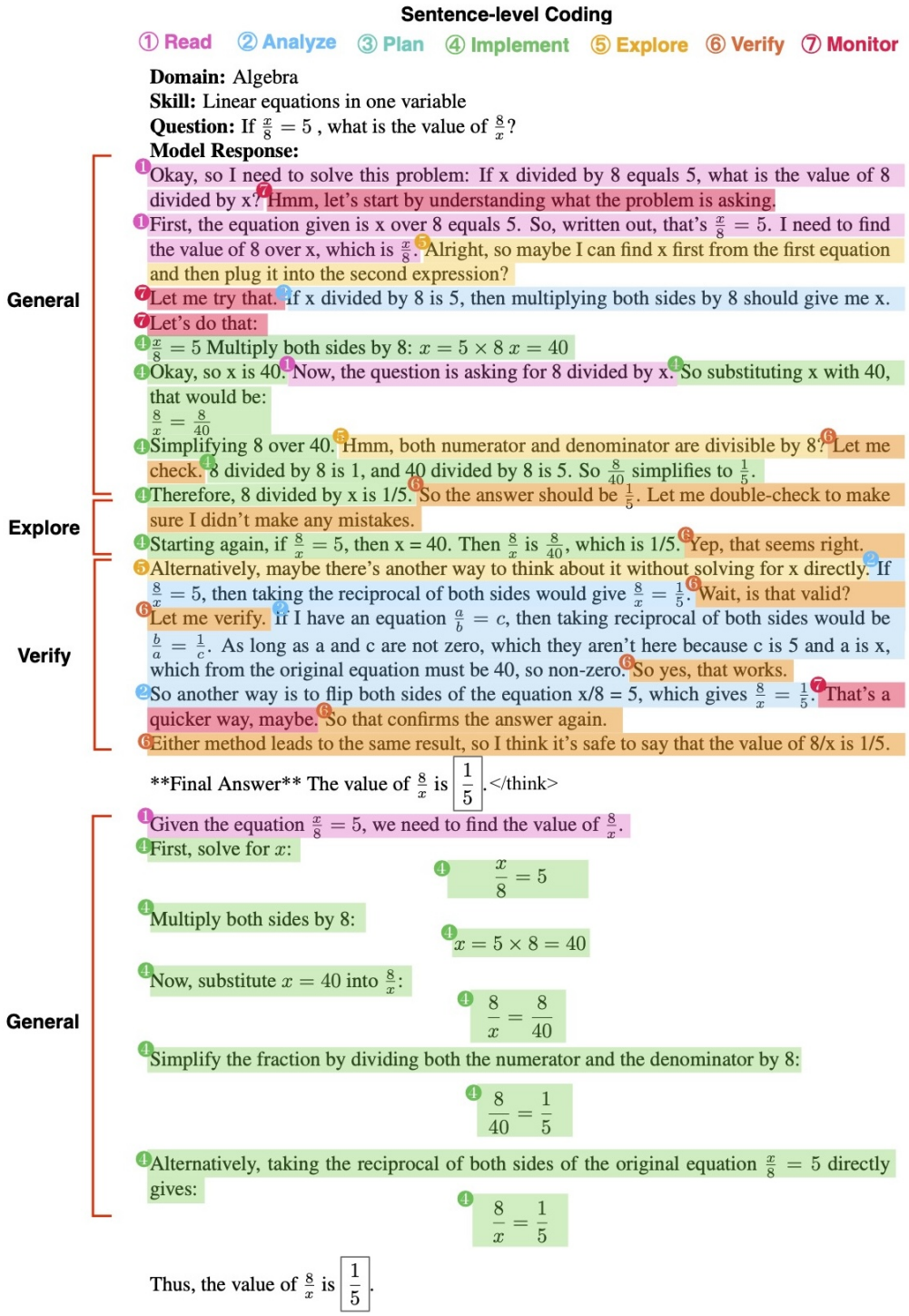

Figure B.1: This figure presents an example of an annotated reasoning process under the adapted Schoenfeld's Episode Theory. Paragraph-level annotations are indicated by square brackets on the left, while sentence-level annotations are color-coded by cognitive process categories.

Schoenfeld [87] 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 [44]. 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. [56] 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.

C.1.2 Large Reasoning Models

The surge in LLM development has catalyzed substantial efforts to bolster their reasoning skills [5, 8, 11]. 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 [18, 90, 108]. Furthermore, instruction tuning using rigorously selected, high-quality datasets has proven effective in boosting performance [48, 51, 53, 68, 111]. Moreover, OpenAI [71], Snell et al. [92] 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.

C.1.3 Efficient Reasoning Methods

Overthinking [12, 23] 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 [25, 94] have been proposed to encourage the reasoning models to reason more efficiently. Specifically, L1 [4] proposes a simple reinforcement learning method that optimizes for accuracy and adherence to user-specified length constraints. ThinkPrune [33] offers a simple solution that continuously trains the long-thinking LLMs via reinforcement learning with an added token limit. Arora and Zanette [7] train reasoning models to dynamically allocate inference-time compute based on task complexity. More works [24, 55, 60, 107] 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.

C.1.4 Reasoning Analysis Methods

Chain-of-thoughts (CoT) [104] can significantly elicit the reasoning capabilities of models. Various works have studied the different properties of CoT, including bias [105], faithfulness [45], and redundancy [12]. 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 [39]. Bogdan et al. [9] 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. [26] 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. [56] and Kargupta et al. [41] incorporated

theories from cognitive psychology to label the episodes of CoT in analogy of human problem-solving processes. Our work, built upon [56], is the first to systematically analyze the statistical patterns of reasoning traces grounded in cognitive theories.

C.2 Implementation Details

C.2.1 Full Model List

We conduct our analysis on a variety of reasoning and non-reasoning models. The reasoning models include DeepSeek-R1 [20], DeepSeek-R1-Distill-Qwen, QwQ-32B [99], Phi4 [2], Qwen3-32B [83], Gemini-2.5-Flash [16], GPT-o1-mini [73] and GPT-o3-mini [77]. The non-reasoning models include GPT-4o [78], Gemini-2.0-Flash [95], Qwen2.5-32B [98] and the non-reasoning mode of Phi4 and Qwen3-32B. We also study some efficient reasoning models including L1 [3], ThinkPrune [33], and models released by Arora and Zanette [7].

C.2.2 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:

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

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.

C.2.3 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:

$$\mathrm{MI}(g) = \sum_{i,j} p(x_i, y_j) \log_2 \frac{p(x_i, y_j)}{p(x_i) \cdot p(y_j)} \tag{C.2}$$

where $x_i \in \{\text{present}, \text{absent}\}$ indicates whether n-gram g appears in a sequence, $y_j \in \{\text{Group 1}, \text{Group 2}\}$ denotes the model group membership, and $p(x_i, y_j)$, $p(x_i)$, and $p(y_j)$ are the joint and marginal probabilities computed from the 2×2 contingency table of n-gram occurrences across the two groups.

C.3 Episode-level Diagnostics of Correctness Details

In this section, we provide the detailed implementation of the correctness diagnostic analysis presented in Section 4.4.1. 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.

C.3.1 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 $e_i \in \mathscr{E}$ and a token count t_i. The set of episode categories $\mathscr{E}$ 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 $c \in \mathscr{E}$, 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*).

C.3.2 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:

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

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.

C.3.3 Full Feature Coefficients.

Table C.1 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 C.1: The full predictive episode-level features for correctness in the diagnostic Lasso logistic regression case study. Features are ranked by their coefficient (β) magnitude.

C.4 Detailed ThinkARM Framework

The detailed ThinkARM Framework is shown in Figure C.1. 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 C.5 and Appendix C.6.

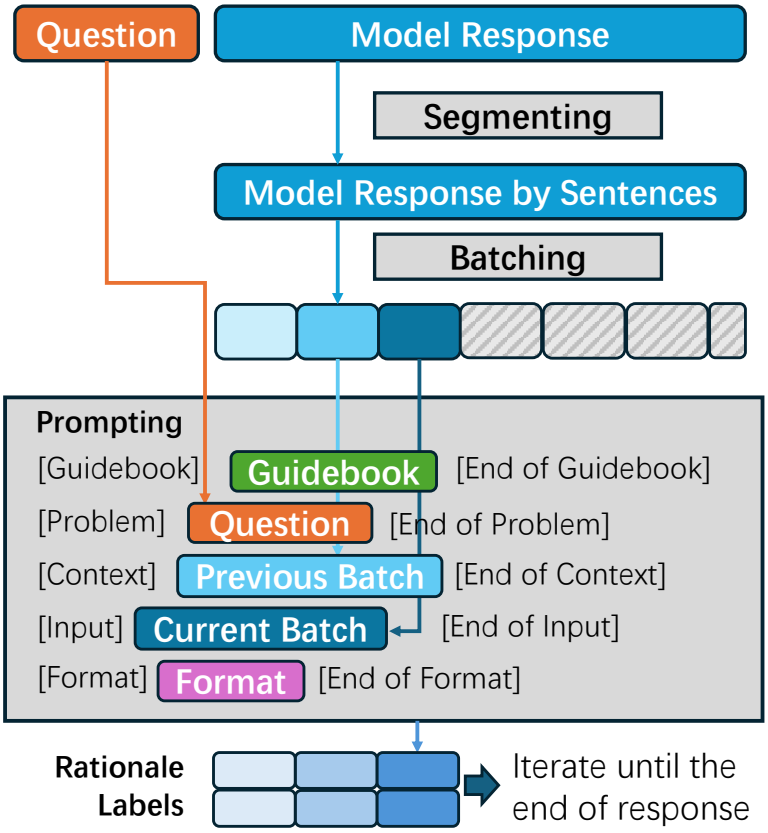

Figure C.1: The ThinkARM Framework. For each question-response pair, the model response is first segmented into sentences. They are then tagged by the annotation models in batches, along with information about the guidebook, question, context, and format. The guidebook is in Appendix C.5, the prompt template is in Appendix C.6.

C.5 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.

C.5.1 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.”

C.5.2 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.

C.6 Annotation Prompt

We use the prompt template in Figure C.2 to annotate the reasoning traces in our study. The detailed content of the guidebook can be found in Appendix C.5.

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'},
...
]}"
[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 C.2: The prompt template we used to annotate the reasoning episode.

Bibliography

[1] Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, and etc. Phi-3 technical report: A highly capable language model locally on your phone, 2024. URL https://arxiv.org/abs/2404.14219.

[2] 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, Adil Salim, Shital Shah, Xin Wang, Rachel Ward, Yue Wu, Dingli Yu, Cyril Zhang, and Yi Zhang. Phi-4 technical report. https://arxiv.org/abs/2412.08905, 2024. arXiv preprint 2412.08905, Dec 2024.

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

[4] Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning, 2025. URL https://arxiv.org/abs/2503.04697.

[5] Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. Large language models for mathematical reasoning: Progresses and challenges. In Neele Falk, Sara Papi, and Mike Zhang, editors, *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, March 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.eacl-srw.17/.

[6] Alfonso Amayuelas, Kyle Wong, Liangming Pan, Wenhu Chen, and William Wang. Knowledge of knowledge: Exploring known-unknowns uncertainty with large language models, 2024. URL https://arxiv.org/abs/2305.13712.

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

[8] 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. Reasoning language models: A blueprint, 2025. URL https://arxiv.org/abs/2501.11223.

[9] Paul C. Bogdan, Uzay Macar, Neel Nanda, and Arthur Conmy. Thought anchors: Which llm reasoning steps matter?, 2025. URL https://arxiv.org/abs/2506.19143.

[10] Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling, 2024. URL https://arxiv.org/abs/2407.21787.

[11] Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models, 2025. URL https://arxiv.org/abs/2503.09567.

[12] 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. Do not think that much for 2+3=? on the overthinking of o1-like llms, 2025. URL https://arxiv.org/abs/2412.21187.

[13] Hyung Won Chung, Le Hou, S. Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Wei Yu, Vincent Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed Huai hsin Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models. *ArXiv*, abs/2210.11416, 2022. URL https://api.semanticscholar.org/CorpusID:253018554.

[14] 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. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[15] Jeremy R. Cole, Michael J. Q. Zhang, Daniel Gillick, Julian Martin Eisenschlos, Bhuwan Dhingra, and Jacob Eisenstein. Selectively answering ambiguous questions, 2023. URL https://arxiv.org/abs/2305.14613.

[16] 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. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025. URL https://arxiv.org/abs/2507.06261.

[17] Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, Nicholas Thumiger, Aditya Desai, Ion Stoica, Ana Klimovic, Graham Neubig, and Joseph E. Gonzalez. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks, 2025. URL https://arxiv.org/abs/2502.08235.

[18] 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, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards, 2025. URL https://arxiv.org/abs/2502.01456.

[19] Ellie Darlington. The use of bloom's taxonomy in advanced mathematics questions. In *Proceedings of the British Society for Research into Learning Mathematics*, volume 33, pages 7–12, 2013.

[20] 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. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

[21] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/.

[22] Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025. URL https://github.com/huggingface/open-r1.

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

[24] Gongfan Fang, Xinyin Ma, and Xinchao Wang. Thinkless: Llm learns when to think, 2025. URL https://arxiv.org/abs/2505.13379.

[25] Sicheng Feng, Gongfan Fang, Xinyin Ma, and Xinchao Wang. Efficient reasoning models: A survey, 2025. URL https://arxiv.org/abs/2504.10903.

[26] Yunzhen Feng, Julia Kempe, Cheng Zhang, Parag Jain, and Anthony Hartshorn. What characterizes effective reasoning? revisiting length, review, and structure of cot, 2025. URL https://arxiv.org/abs/2509.19284.

[27] Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D. Goodman. Stream of search (sos): Learning to search in language, 2024. URL https://arxiv.org/abs/2404.03683.

[28] Kanishk Gandhi, Abhishek Chakravarthy, Ansh Singh, Nik Lile, and Noah D. Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars, 2025. URL https://arxiv.org/abs/2503.01307.

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

[30] Carole Greenes. Mathematics learning and knowing: A cognitive process. *Journal of Education*, 177(1):85–106, 1995. doi: 10.1177/002205749517700105.

[31] E. Harskamp and C. Suhre. Schoenfeld's problem solving theory in a student controlled learning environment. *Computers & Education*, 49(3):822–839, November 2007. doi: 10.1016/j.compedu.2005.11.024. URL https://doi.org/10.1016/j.compedu.2005.11.024.

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

[33] Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2504.01296.

[34] Zhenyu Hou, Xin Lv, Rui Lu, Jiajie Zhang, Yujiang Li, Zijun Yao, Juanzi Li, Jie Tang, and Yuxiao Dong. Advancing language model reasoning through reinforcement learning and inference scaling, 2025. URL https://arxiv.org/abs/2501.11651.

[35] Jie Huang and Kevin Chen-Chuan Chang. Towards reasoning in large language models: A survey, 2023. URL https://arxiv.org/abs/2212.10403.

[36] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55, January 2025. ISSN 1558-2868. doi: 10.1145/3703155. URL http://dx.doi.org/10.1145/3703155.

[37] Lulu Huang, Da Li, Hao Liu, and Lingxiao Cheng. Beyond accuracy: The role of calibration in self-improving large language models. *arXiv preprint arXiv:2504.02902*, 2025. URL https://arxiv.org/abs/2504.02902.

[38] 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. Mme-cot: Benchmarking chain-of-thought in large multimodal models for reasoning quality, robustness, and efficiency, 2025. URL https://arxiv.org/abs/2502.09621.

[39] Gangwei Jiang, Yahui Liu, Zhaoyi Li, Qi Wang, Fuzheng Zhang, Linqi Song, Ying Wei, and Defu Lian. What makes a good reasoning chain? uncovering structural patterns in long chain-of-thought reasoning, 2025. URL https://arxiv.org/abs/2505.22148.

[40] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

[41] 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. Cognitive foundations for reasoning and their manifestation in llms, 2025. URL https://arxiv.org/abs/2511.16660.

[42] David R. Krathwohl. A revision of bloom's taxonomy: An overview. *Theory into Practice*, 41 (4):212–218, 2002. doi: 10.1207/s15430421tip4104_2.

[43] Abhinav Kumar, Jaechul Roh, Ali Naseh, Marzena Karpinska, Mohit Iyyer, Amir Houmansadr, and Eugene Bagdasarian. Overthink: Slowdown attacks on reasoning llms, 2025. URL https://arxiv.org/abs/2502.02542.

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

[45] 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, Saurav Kadavath, Shannon Yang, Thomas Henighan, Timothy Maxwell, Timothy Telleen-Lawton, Tristan Hume, Zac Hatfield-Dodds, Jared Kaplan, Jan Brauner, Samuel R. Bowman, and Ethan Perez. Measuring faithfulness in chain-of-thought reasoning, 2023. URL https://arxiv.org/abs/2307.13702.

[46] Kuang-Huei Lee, Ian Fischer, Yueh-Hua Wu, Dave Marwood, Shumeet Baluja, Dale Schuurmans, and Xinyun Chen. Evolving deeper llm thinking, 2025. URL https://arxiv.org/abs/2501.09891.

[47] Noam Levi. A simple model of inference scaling laws, 2024. URL https://arxiv.org/abs/2410.16377.

[48] Ming Li, Lichang Chen, Jiuhai Chen, Shwai He, Jiuxiang Gu, and Tianyi Zhou. Selective reflection-tuning: Student-selected data recycling for LLM instruction-tuning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics ACL 2024*, pages 16189–16211, Bangkok, Thailand and virtual meeting, August 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.findings-acl.958.

[49] Ming Li, Yong Zhang, Shwai He, Zhitao Li, Hongyu Zhao, Jianzong Wang, Ning Cheng, and Tianyi Zhou. Superfiltering: Weak-to-strong data filtering for fast instruction-tuning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14255–14273, Bangkok, Thailand, August 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.acl-long.769.

[50] Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. From quantity to quality: Boosting LLM performance with self-guided data selection for instruction tuning. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *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, June 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.naacl-long.421.

[51] Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. From quantity to quality: Boosting LLM performance with self-guided data selection for instruction tuning. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *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, June 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.naacl-long.421.

[52] Ming Li, Chenrui Fan, Yize Cheng, Soheil Feizi, and Tianyi Zhou. Schoenfeld's anatomy of mathematical reasoning by language models. *arXiv preprint*, 2025.

[53] Ming Li, Yanhong Li, Ziyue Li, and Tianyi Zhou. How instruction and reasoning data shape post-training: Data quality through the lens of layer-wise gradients. *arXiv preprint arXiv:2504.10766*, 2025.

[54] Ming Li, Yanhong Li, and Tianyi Zhou. What happened in LLMs layers when trained for fast vs. slow thinking: A gradient perspective. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32017–32154, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. URL https://aclanthology.org/2025.acl-long.1545/.

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

[56] Ming Li, Nan Zhang, Chenrui Fan, Hong Jiao, Yanbin Fu, Sydney Peters, Qingshu Xu, Robert Lissitz, and Tianyi Zhou. Understanding the thinking process of reasoning models: A perspective from schoenfeld's episode theory. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 18278–18299, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.922. URL https://aclanthology.org/2025.emnlp-main.922/.

[57] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

[58] Bill Yuchen Lin, Abhilasha Ravichander, Ximing Lu, Nouha Dziri, Melanie Sclar, Khyathi Chandu, Chandra Bhagavatula, and Yejin Choi. The unlocking spell on base llms: Rethinking alignment via in-context learning. *arXiv preprint arXiv:2312.01552*, 2023.

[59] Changshu Liu, Shizhuo Dylan Zhang, Ali Reza Ibrahimzada, and Reyhaneh Jabbarvand. Codemind: A framework to challenge large language models for code reasoning, 2024. URL https://arxiv.org/abs/2402.09664.

[60] Wei Liu, Ruochen Zhou, Yiyun Deng, Yuzhen Huang, Junteng Liu, Yuntian Deng, Yizhe Zhang, and Junxian He. Learn to reason efficiently with adaptive length-based reward shaping, 2025. URL https://arxiv.org/abs/2505.15612.

[61] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. URL https://arxiv.org/abs/1907.11692.

[62] Yue Liu, Jiaying Wu, Yufei He, Hongcheng Gao, Hongyu Chen, Baolong Bi, Jiaheng Zhang, Zhiqi Huang, and Bryan Hooi. Efficient inference for large reasoning models: A survey, 2025. URL https://arxiv.org/abs/2503.23077.

[63] Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning, 2025. URL https://arxiv.org/abs/2501.12570.

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

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

[66] Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. Cross-task generalization via natural language crowdsourcing instructions. *arXiv preprint arXiv:2104.08773*, 2021.

[67] Melanie Mitchell. Artificial intelligence learns to reason. *Science*, 387(6740):eadw5211, 2025. doi: 10.1126/science.adw5211. URL https://www.science.org/doi/10.1126/science.adw5211.

[68] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling, 2025. URL https://arxiv.org/abs/2501.19393.

[69] Sam Musker, Alex Duchnowski, Raphaël Millière, and Ellie Pavlick. Llms as models for analogical reasoning, 2025. URL https://arxiv.org/abs/2406.13803.

[70] OpenAI. Gpt-4 technical report, 2023.

[71] OpenAI. Learning to reason with llms, 2024. URL https://openai.com/index/learning-to-reason-with-llms/.

[72] OpenAI. OpenAI o1 System Card, December 2024. URL https://cdn.openai.com/o1-system-card-20241205.pdf.

[73] OpenAI. OpenAI o1-mini System Card, September 2024. URL https://openai.com/index/openai-o1-mini-advancing-cost-efficient-reasoning/.

[74] OpenAI. Openai o1 system card. https://openai.com/index/openai-o1-system-card/, 2024. Accessed: 2025-08-12.

[75] OpenAI. Introducing GPT-4.1 in the API, April 2025. URL https://openai.com/index/gpt-4-1/.

[76] OpenAI. OpenAI GPT-5 System Card, August 2025. URL https://openai.com/index/gpt-5-system-card/.

[77] OpenAI. OpenAI o3-mini System Card, January 2025. URL https://cdn.openai.com/o3-mini-system-card-feb10.pdf.

[78] 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. Gpt-4o system card, 2024. URL https://arxiv.org/abs/2410.21276.

[79] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, and etc. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

[80] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.168. URL https://aclanthology.org/2021.naacl-main.168.

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

[82] Xiaoye Qu, Yafu Li, Zhaochen Su, Weigao Sun, Jianhao Yan, Dongrui Liu, Ganqu Cui, Daizong Liu, Shuxian Liang, Junxian He, Peng Li, Wei Wei, Jing Shao, Chaochao Lu, Yue Zhang, Xian-Sheng Hua, Bowen Zhou, and Yu Cheng. A survey of efficient reasoning for large reasoning models: Language, multimodality, and beyond, 2025. URL https://arxiv.org/abs/2503.21614.

[83] Qwen Team. Qwen3: Think deeper, act faster, April 2025. URL https://qwenlm.github.io/blog/qwen3/.

[84] Farzad Radmehr and Michael Drake. Revised bloom's taxonomy and major theories and frameworks that influence the teaching, learning, and assessment of mathematics: a comparison. *International Journal of Mathematical Education in Science and Technology*, 49(6):917–929, 2018. doi: 10.1080/0020739X.2018.1549336. URL https://www.tandfonline.com/doi/abs/10.1080/0020739X.2018.1549336.

[85] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410/.

[86] Teven Le Scao, Angela Fan, Christopher Akiki, Elizabeth-Jane Pavlick, Suzana Ili'c, Daniel Hesslow, Roman Castagn'e, Alexandra Sasha Luccioni, Franccois Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Rose Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurenccon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa Etxabe, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu,

Chenghao Mou, Chris C. Emezue, Christopher Klamm, Colin Leong, Daniel Alexander van Strien, David Ifeoluwa Adelani, Dragomir R. Radev, Eduardo Gonz'alez Ponferrada, Efrat Levkovizh, Ethan Kim, Eyal Bar Natan, Francesco De Toni, Gérard Dupont, Germán Kruszewski, Giada Pistilli, Hady ElSahar, Hamza Benyamina, Hieu Trung Tran, Ian Yu, Idris Abdulmumin, Isaac Johnson, Itziar Gonzalez-Dios, Javier de la Rosa, Jenny Chim, Jesse Dodge, Jian Zhu, Jonathan Chang, Jorg Frohberg, Josephine L. Tobing, Joydeep Bhattacharjee, Khalid Almubarak, Kimbo Chen, Kyle Lo, Leandro von Werra, Leon Weber, Long Phan, Loubna Ben Allal, Ludovic Tanguy, Manan Dey, Manuel Romero Muñoz, Maraim Masoud, Mar'ia Grandury, Mario vSavsko, Max Huang, Maximin Coavoux, and Mayank Singh. Bloom: A 176b-parameter open-access multilingual language model. *ArXiv*, abs/2211.05100, 2022. URL https://api.semanticscholar.org/CorpusID:253420279.

[87] Alan H. Schoenfeld. *Mathematical Problem Solving*. Academic Press, 1985.

[88] Alan H. Schoenfeld. Learning to think mathematically: Problem solving, metacognition, and sense making in mathematics. *Journal of Education*, 196(2):1–38, 2016.

[89] Lianlei Shan, Shixian Luo, Zezhou Zhu, Yu Yuan, and Yong Wu. Cognitive memory in large language models, 2025. URL https://arxiv.org/abs/2504.02441.

[90] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

[91] Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, and Shiguo Lian. Dast: Difficulty-adaptive slow-thinking for large reasoning models, 2025. URL https://arxiv.org/abs/2503.04472.

[92] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

[93] Mark Steyvers and Megan A. K. Peters. Metacognition and uncertainty communication in humans and large language models. *arXiv preprint arXiv:2504.14045*, 2025. URL https://arxiv.org/abs/2504.14045.

[94] Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, and Xia Hu. Stop overthinking: A survey on efficient reasoning for large language models, 2025. URL https://arxiv.org/abs/2503.16419.

[95] 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 etc. Gemini: A family of highly capable multimodal models, 2024. URL https://arxiv.org/abs/2312.11805.

[96] Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, Soroosh Mariooryad, Yifan Ding, Xinyang Geng, and etc. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context, 2024. URL https://arxiv.org/abs/2403.05530.

[97] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, and etc. Gemma 2: Improving open language models at a practical size, 2024. URL https://arxiv.org/abs/2408.00118.

[98] Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

[99] Qwen Team. Qwq-32b: Embracing the power of reinforcement learning, March 2025. URL https://qwenlm.github.io/blog/qwq-32b/.

[100] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[101] Yaoting Wang, Shengqiong Wu, Yuecheng Zhang, Shuicheng Yan, Ziwei Liu, Jiebo Luo, and Hao Fei. Multimodal chain-of-thought reasoning: A comprehensive survey, 2025. URL https://arxiv.org/abs/2503.12605.

[102] Yuqing Wang and Yun Zhao. Gemini in reasoning: Unveiling commonsense in multimodal large language models, 2023. URL https://arxiv.org/abs/2312.17661.

[103] Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=gEZrGCozdqR.

[104] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

[105] Xuyang Wu, Jinming Nian, Ting-Ruen Wei, Zhiqiang Tao, Hsin-Tai Wu, and Yi Fang. Does reasoning introduce bias? a study of social bias evaluation and mitigation in llm reasoning, 2025. URL https://arxiv.org/abs/2502.15361.

[106] Shijie Xia, Xuefeng Li, Yixin Liu, Tongshuang Wu, and Pengfei Liu. Evaluating mathematical reasoning beyond accuracy, 2025. URL https://arxiv.org/abs/2404.05692.

[107] Violet Xiang, Chase Blagden, Rafael Rafailov, Nathan Lile, Sang Truong, Chelsea Finn, and Nick Haber. Just enough thinking: Efficient reasoning with adaptive length penalties reinforcement learning, 2025. URL https://arxiv.org/abs/2506.05256.

[108] Wei Xiong, Hanning Zhang, Chenlu Ye, Lichang Chen, Nan Jiang, and Tong Zhang. Self-rewarding correction for mathematical reasoning, 2025. URL https://arxiv.org/abs/2502.19613.

[109] Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models, 2024. URL https://arxiv.org/abs/2402.13116.

[110] Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. Hallucination is inevitable: An innate limitation of large language models, 2025. URL https://arxiv.org/abs/2401.11817.

[111] Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning, 2025. URL https://arxiv.org/abs/2502.03387.

[112] Joseph B. W. Yeo and Ban Har Yeap. Characterising the cognitive processes in mathematical investigation. *International Journal for Mathematics Teaching and Learning*, pages 1–10, 2010. URL http://www.cimt.org.uk/journal/jbwyeo.pdf.

[113] Weixiang Zhao, Xingyu Sui, Jiahe Guo, Yulin Hu, Yang Deng, Yanyan Zhao, Bing Qin, Wanxiang Che, Tat-Seng Chua, and Ting Liu. Trade-offs in large reasoning models: An empirical analysis of deliberative and adaptive reasoning over foundational capabilities. *arXiv preprint arXiv:2503.17979*, 2025. URL https://arxiv.org/abs/2503.17979.

[114] Kaitlyn Zhou, Dan Jurafsky, and Tatsunori Hashimoto. Navigating the grey area: How expressions of uncertainty and overconfidence affect language models, 2023. URL https://arxiv.org/abs/2302.13439.