Why Do Reasoning Models Lose Coverage? The Role of Data and Forks in the Road

Ngoc-Hieu Nguyen[*]
The Pennsylvania State University
hnn5071@psu.edu

Parshin Shojaee[*]
Virginia Tech

Phuc Minh Nguyen
VinUniversity

Nan Zhang
The Pennsylvania State University

Chandan K Reddy
Virginia Tech

Khoa D Doan[†]
VinUniversity

Rui Zhang[†]
The Pennsylvania State University

Abstract

Recent progress in large language models has led to the emergence of reasoning models, which have shown strong performance on complex tasks through specialized fine-tuning procedures. While these methods reliably improve pass@1 accuracy, prior works have observed that they show a coverage shrinkage behavior, where pass@k degrades relative to the base model. In this paper, we investigate the reasoning shrinkage arise under SFT-based post-training. We hypothesize that this behavior is driven by properties of the fine-tuning data, specifically related to decision points or "forks in the road" scenarios where model faces indecipherable patterns with multiple valid reasoning paths. To test this hypothesis, we design controlled case studies that simulate such decision-point settings, spanning indecipherable nodes in graph branching, and reasoning modes. By tracking post-training dynamics in these settings, we find that the shrinkage phenomenon is tightly correlated with the prevalence of decision-point scenarios in the training data. We also demonstrate that this shrinkage behavior can be partially mitigated through targeted data synthesis design of decision-points, and a more systematic diversity-encouraging decoding mechanism. Our findings identify data-centric factors as a key driver of shrinkage in reasoning models and highlight diversity-aware designs as an effective lever for controlling it.[1]

"For every path you choose, there is another you must abandon"
— Joan D. Vinge

1 Introduction

Large language models (LLMs) fine-tuned for reasoning have recently achieved remarkable gains on complex reasoning benchmarks. Through post-training procedures such as reasoning-based supervised fine-tuning (SFT) and reinforcement learning with verification feedback (RLVR) (Lambert et al., 2025; Guo et al., 2025), these models reliably have shown improvement on pass@1 accuracy, often substantially outperforming their base counterparts. As a result, reasoning-oriented post-training has become a central paradigm for improving the capabilities of LLMs on challenging tasks.

[*]Equal contribution.
[†]Equal advising.
[1]Data and code for reproducing our experiments are available at: https://github.com/psunlpgroup/reasoning_forks

Despite these gains, a growing body of work in literature has identified a concerning and counterintuitive phenomenon known as *coverage shrinkage*, where improvements in pass@1 are accompanied by a degradation in pass@k, indicating a loss of diversity and coverage in the model's learned reasoning procedures. Understanding why this happens, and what fundamentally drives this shrinkage, remains an open and important research question. Most prior works have attributed this behavior to the optimization dynamics of RLVR (Yue et al., 2025; Wu & Choi, 2025; Liu et al., 2025; Shojaee et al., 2025), arguing that reward-driven training overemphasizes a narrow set of high-reward reasoning paths and leads to an increased chance of mode collapse. However, recent works have shown that coverage shrinkage is not unique to RLVR and also arises under supervised post-training with SFT (Chen et al., 2025; Dang et al., 2025), which implies that shrinkage may not simply be a consequence of post-training algorithms, but may arise from a more fundamental driver.

In this paper, we argue that data plays a central and underexplored role in driving the coverage shrinkage behavior. Instead of focusing on post-training algorithms, we shift attention to the structure and properties of the post-training data, and ask a simple but critical question: *what aspects of reasoning data encourage the models to collapse onto fewer solution paths?* Our key hypothesis is that shrinkage is strongly influenced by *decision points* or *"forks in the road"* in the data, situations in which a model encounters multiple valid, indistinguishable reasoning paths and must commit to one. At such points, post-training implicitly pressures the model to commit to a subset of these options available. Over time, this commitment suppresses alternative trajectories, resulting in improved correctness along dominant paths but reduced overall coverage.

Figure 1: Reasoning unfolds through *forks in the road*, choices made without knowing the path to truth.

To test this hypothesis, we design controlled case studies that isolate and expose these decision-point structures to the reasoning (Figure 2). Our first setting is a graph-based navigation task, a natural testbed for studying decision points, which is inspired by prior work on *indecipherable nodes* in next-token prediction (Bachmann & Nagarajan, 2024). In this task, a model must traverse a star graph from a start node to a target node while encountering the branching points that provide no information about which branch leads to success. Our second setting focuses on mathematical reasoning problems that admit multiple valid reasoning modes and solution strategies. At certain stages of reasoning, the model must decide how to proceed without knowing which strategy will ultimately succeed; these moments constitute decision points analogous to the graph branching. By tracking post-training dynamics across both of these settings as well as across ablated data variants, we observe that post-training coverage shrinkage is largely driven by how models resolve ambiguity under repeated exposure to such decision points. In particular, ablating or restructuring these points significantly alters model behavior, and the degree of shrinkage is correlated with their prevalence in the post-training data.

Motivated by our findings, we introduce two practical strategies to control such shrinkage phenomenon. First, we show that per-problem coverage of alternative decisions can significantly control shrinkage behavior compared to distributing diversity across the problems. This highlights the importance of synthetic data design that accounts for decision-point structure and its coverage within problems. Second, we observe that shrinkage is closely associated with the emergence of high-frequency tokens that often correspond to early, indecipherable choices for the model in the reasoning process. We demonstrate that a simple decoding mechanism that encourages diversity among these dominant initial tokens can partially recover the lost coverage after the post-training without the need for any additional re-training. Our findings also help explain why building a unified model that can

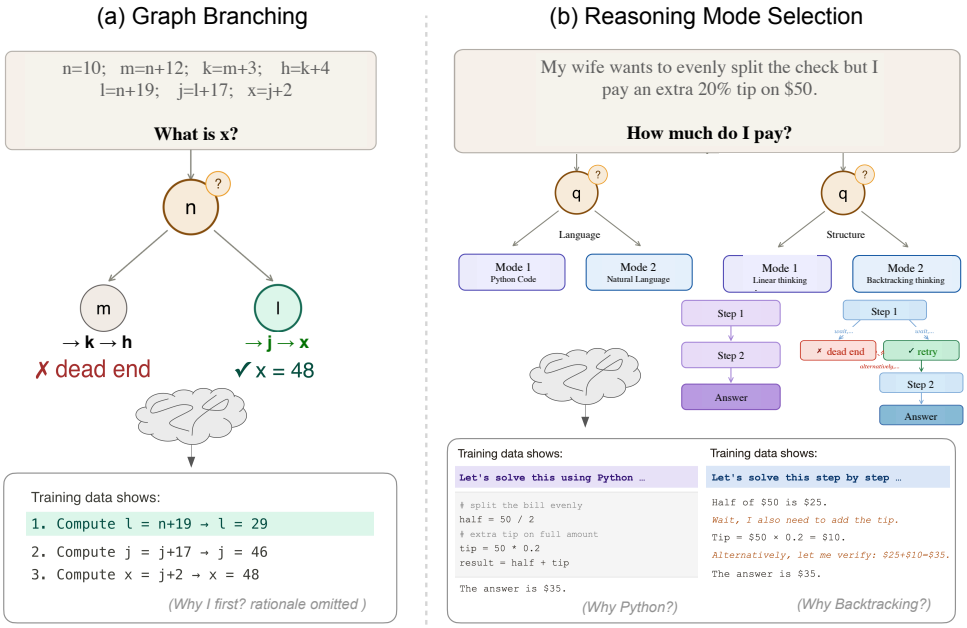


Figure 2: **Illustrative examples of *forks in the road* case studies.** **(a)** Graph navigation with indecipherable nodes; and **(b)** Mathematical reasoning with multiple valid solution modes. In both settings, decision points force commitment to a path without knowing which will succeed.

effectively operate in both instruct (or non-thinking) and backtracking/thinking reasoning modes remains a challenging open problem (Wang et al., 2025).

This study is mainly aimed to deepen our understanding of coverage shrinkage in reasoning models and provides a data-centric perspective on this phenomenon. Our key contributions are as follows:

- We present a systematic, *data-centric* study of coverage shrinkage in reasoning post-trained models, aiming to understand its underlying factors.
- We identify forks-in-the-road patterns in fine-tuning data as a key driver of coverage shrinkage, and analyze this effect through targeted case studies such as graph branching and alternative mathematical reasoning strategies.
- Through controlled experiments and training-dynamics analysis, we find a strong correlation between the structure of decision points in data and the severity of coverage shrinkage, providing empirical evidence for the role of data on such behavior.
- Motivated by our findings, we introduce two simple diversity-aware data synthesis and decoding strategies, and present proof-of-concept results demonstrating their effectiveness in mitigating shrinkage. These results suggest that the lost coverage is not permanently forgotten, but instead suppressed, and can be recovered through effective inference-time intervention.

2 Preliminaries

To formally ground our investigation, we first define the standard metrics and review the mechanics of SFT and RLVR post-training.

Coverage Definition. Coverage is usually the primary metric for assessing the effectiveness of test-time scaling (Brown et al., 2025). It represents the highest performance achievable when solutions can be sampled repeatedly. Let $R(x, y) \in 0, 1$ be a binary reward indicating whether y correctly solves x. Let $\pi(\cdot \mid x)$ denote the model's output distribution given x, and $D(\pi(\cdot \mid x))$ its decoding distribution. The pass@k metric for x is the probability that at least one correct solution appears in k independent samples:

pass@k$(x) = 1 - (1 - \text{pass@1})^k$, where pass@1 $= \mathbb{P}_{y \sim D(\pi(\cdot|x))}[R(x, y) = 1]$. It captures the effectiveness of repeated sampling when automatic verification is available (Brown et al., 2025). In practice, pass@k could be estimated from $n \geq k$ samples per problem $x_i \in \mathcal{D}_X$, as pass@$k := \frac{1}{|\mathcal{D}_X|} \sum_{x_i \in \mathcal{D}_X} \left[1 - \binom{n-c_i}{k} / \binom{n}{k}\right]$ where c_i is the number of correct samples. In the setting of generation with chain-of-thought (CoT), we sample k reasoning chains $\tau_i^{(k)} \sim D(\pi(\cdot|x_i))$ and corresponding final answers $y_i^{(k)} \sim D(\pi(\cdot|\tau_i^{(k)}, x_i))$. These chains may differ in phrasing and format (Zhao et al., 2025) or in the reasoning skills they invoke (He et al., 2025), giving rise to exploration of reasoning at test-time.

Post-training and Coverage Shrinkage. In practice, to enhance the reasoning capabilities of LLMs, a common approach is to first perform reasoning-based fine-tuning with SFT on a dataset of triples (x_i, τ_i, y_i), where x_i denotes a problem instance, τ_i is a corresponding reasoning chain, and y_i is the final answer. For most problems x_i, the dataset contains only one or a small number of annotated reasoning chains τ_i. As a result, SFT encourages the model to concentrate probability mass on this limited set by maximizing $\pi_\theta(y_i, \tau_i \mid x_i)$. This training procedure typically improves pass@1 by increasing the likelihood of generating reasoning chains that closely match the annotated τ_i and y_i for each x_i in the training set. Following SFT, the reinforcement learning with verifiable rewards (RLVR) (Lambert et al., 2025; Guo et al., 2025) is often applied as another post-training stage, and has demonstrated substantial empirical gains on verifiable reasoning tasks, including mathematics and coding. Recent analyses have examined both the strengths and limitations of these post-training stages for reasoning, identifying a counterintuitive phenomenon termed as *sharpening* or *coverage shrinkage* where improvements in pass@1 often coincide with declines in pass@k (Dang et al., 2025; Chen et al., 2025; Yue et al., 2025; Wu & Choi, 2025). This effect is also commonly attributed to reduced diversity of the model's reasoning paths. In our paper, we primarily seek to understand this coverage shrinkage behavior and provide a data-centric perspective on its underlying factors.

3 A Data-Centric View: Forks in the Road

Solving reasoning problems is inherently exploratory: one often encounters multiple candidate strategies without knowing which will succeed and must decide a path forward. Strong problem solvers are not just those who can execute a given direction, but those who can think about diverse approaches, evaluate them, and select the most promising ones.

Nevertheless, current reasoning-focused fine-tuning datasets often suffer from survivorship bias: models are typically exposed only to the final, successful reasoning path during post-training. Specifically, in tasks with multiple valid reasoning paths, the model is mostly exposed to one of these paths per problem, hiding the alternative strategies and the rationale behind the choice of the given reasoning path. This "missing rationale" problem manifests at both micro-level (step-by-step or which algebraic manipulation to apply next) and macro-level (strategy or mode selection). For instance, the model may need to choose between distinct solving approaches like analytical natural language reasoning versus directly writing a Python script. However, provided ground truth answers do not explain how or why this choice is made; instead, the final decision is simply reflected in the data presented to the model. At such decision points, when the model is exposed to a single reasoning path without justification, it may struggle to make decisions under uncertainty and instead rely on spurious cues that steer it toward a particular path or mode of the reasoning.

Synthesizing these observations and our empirical results, we propose the following hypothesis regarding the failure modes of reasoning models:

Fork-in-the-road Hypothesis

When the solution space contains multiple viable paths but the training data obscures the rationale for selecting among them—particularly at decision points or "forks in the road"—the model fails to learn a generalizable selection mechanism. Instead, it becomes implicitly pressured to commit to a subset of available paths, relying on spurious, non-causal cues to resolve ambiguity. Over time, this leads to the suppression of alternative trajectories, resulting in coverage collapse at inference time, as evidenced by degraded *pass@k* performance after post-training.

To test this hypothesis, we design controlled case studies, as shown in Figure 2, that isolate and highlight these decision-point structures within the SFT data, as detailed in the following sections.

4 Experimental Framework

4.1 Graph Navigation

Studying the role of decision points in sophisticated reasoning tasks like mathematical problem solving is not trivial due to the presence of multiple confounding factors, such as problem complexity, heterogeneous solution strategies, and implicit forms of supervision. To enable controlled evaluation of our hypothesis, we begin with a simple synthetic setting that isolates the role of such decision points in reasoning processes (shown in Figure 2). Inspired by Zhang et al. (2022) and Bachmann & Nagarajan (2024), we focus experiments on a dependency chain evaluation task with a star-graph structure, a natural testbed for decision points. Concretely, each prompt specifies functional dependencies among variables (graph nodes) and a query over a target variable. Figure 2 (left) shows an example with its ground-truth solution. Due to the binary branching nodes, the first step (highlighted in green) is a decision point where both m and l are valid continuations; however, only l leads to the correct target x. While this can be determined via search, the model is only exposed to the successful trajectory at training, leaving the underlying decision process implicit. This design allows us to control and ablate the presence of decision points in the solutions while holding other factors fixed.

To isolate the role of decision points in coverage shrinkage analysis, we construct two controlled data variants from the same synthetic task that differ only in whether such decision points are present during post-training. In the *Forward* setting, the model encounters explicit decision points—states with multiple possible continuations where only one leads to a correct solution. Solving the problem therefore requires the model to decide and select among reasoning paths. In contrast, the *Reverse (w/o DP)* setting removes these forks by presenting the correct trajectory in reverse format (target to source), effectively eliminating any need for path selection by the model. Because both settings correspond to the same underlying task, any behavioral differences can be directly attributed to the impact of decision points in the data. We evaluate this setup on two backbone

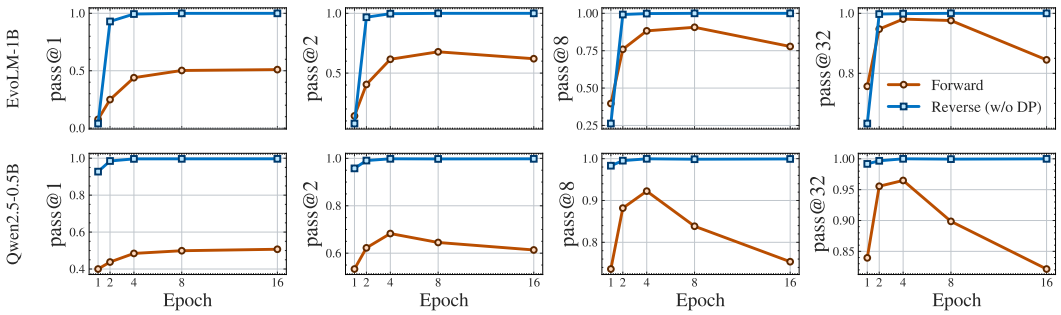


Figure 3: **Effect of decision points on coverage in graph navigation task.** Pass@k across SFT epochs for *Forward* vs. *Reverse* (w/o DP) problem solving settings.

models (`Qwen-2.5-0.5B` (Team, 2024) and `EvoLM-1B` (Qi et al., 2025)) to test robustness of observations with respect to pre-training initialization prior to the post-training.

Results. Figure 3 shows a clear divergence between these two settings. When decision points are removed (*Reverse (w/o DP)*), both models quickly reach near-perfect *pass@k* performance across all values of k, and this performance remains stable even for large k and extended training time. This indicates that the models retain broad coverage over valid solution paths in the reverse reasoning setting, when they are not explicitly exposed to moving froward branching and forced to make decisions during training. However, when decision points are present (*Forward*), the behavior completely changes. While pass@1 improves steadily, performance for larger k follows a different pattern: it first improves, then degrades as training continues. This degradation becomes more pronounced as k increases, providing direct evidence of sharpening and coverage shrinkage behavior. The model becomes narrower, concentrating probability mass on a single preferred path while losing other valid alternatives of graph branching.

We also looked deeper into *Forward* model behavior at decision points (Figure 4). In this setting, we observe that the model's confidence at decision points (measured by token-level probability) increases sharply throughout the epochs of training (**left**). However, this increase is not selective: the model is highly confident not only when it chooses the correct branch, but also when it chooses an incorrect one (**right**). This shows that training with decision points in data can push the model toward overconfident, single-path commitments, rather than calibrated uncertainty of correct solutions over multiple valid continuations. As a result, alternative trajectories are progressively suppressed, leading to the observed coverage shrinkage and drop in pass@k.

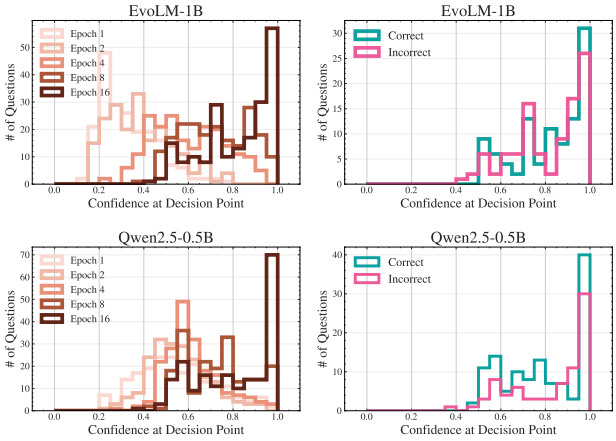


Figure 4: **Left:** Change in model confidence at decision points over the course of SFT; **Right:** At the last epoch, the model assigns high confidence to both correct and incorrect paths, indicating uncalibrated decisions that drive coverage collapse.

We have also validated this hypothesis in other post-training methods, such as RLVR. As shown in Appendix Section A.4, the same coverage shrinkage emerges during RLVR when training on forward vs reverse settings, and is even more pronounced than under SFT post-training. This suggests that coverage shrinkage may not be driven only by the learning algorithm, but also by the data, its structure, and the presence of decision points in reasoning.

Learning Spurious Cues for Branch Selection. Previous work shows that reasoning in language models is highly sensitive to minor input changes (Mirzadeh et al., 2025; Jiang et al., 2024b). We further examine this by probing the model's branch selection: we perturb prompts by shuffling variable dependencies (e.g., `n=10; m=n+12; k=m+3` to `m=n+12; n=10; k=m+3`), preserving semantics but altering surface formats. We observe that despite the same logic, these changes can significantly shift the model's behavior, causing it to select different reasoning branches (as shown in Figure 5). This suggests the model may rely on spurious cues like dependency order when it encounters decision points. We provide more details regarding this fragility in distilled reasoning models and their thinking structure in Section 4.2.2.

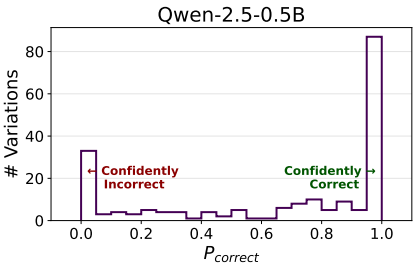


Figure 5: Model confidence at decision points across prompt variations with identical semantics per problem.

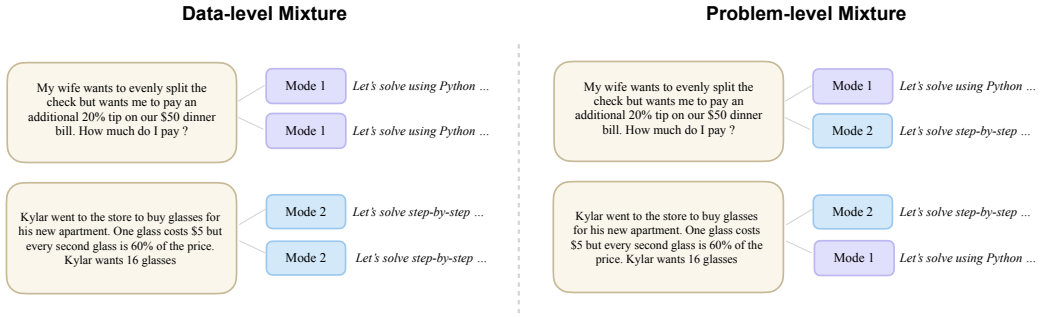


Figure 6: **Diversity structure in SFT data.** Data-level diversity distributes modes across problems, while problem-level diversity exposes multiple reasoning modes within each problem.

4.2 Reasoning Mode Selection

The forks-in-the-road phenomenon is not limited to synthetic graph navigation settings; it also arises naturally in real-world reasoning tasks where multiple solution strategies coexist. During generation, the model must implicitly commit to one early in the trajectory, before knowing which will succeed. These early commitments act as decision points, analogous to branching in graph-based settings. To study this effect, we examine reasoning mode selection as a representative instance of such decision points. Specifically, we consider two settings: (i) different reasoning representations: natural language (NL) versus code-based reasoning, and (ii) different reasoning structures: linear vs. backtracking reasoning. We analyze each of these settings in detail below.

4.2.1 Natural Language vs. Code Reasoning

We treat the choice between natural language (NL) and code-based reasoning as an explicit decision point. Early in generation, the model must commit to one mode, making this setting a natural extension of the fork-in-the-road analysis in Section 4.1. In this section, we investigate whether models trained on mixed data can learn to balance different reasoning modes under repeated sampling. A key question is how the *structure* of diversity in training data affects this decision. In our experiments, we construct two data designs with identical diversity ratios (50% NL, 50% code) but different organization (Figure 6): **Data-level diversity:** each problem is solved using a single mode, but the dataset is globally balanced across the modes; **Problem-level diversity:** each problem appears in fine-tuning data with both reasoning modes. This setup helps us to better understand whether coverage depends on just *how much* diversity is present, or *how it is also distributed*.

Setup. In this analysis, we collect SFT samples from two common mathematical reasoning datasets: OpenMathInstruct-1 (Toshniwal et al., 2024b), and OpenMathInstruct-2 (Toshniwal et al., 2024a). The OpenMathInstruct-2 consist of solutions in natural language, while OpenMathInstruct-1 consists of Python code solutions generated by Mixtral-8x7B (Jiang et al., 2024a). We first filter only GSM8k questions and combine them from these two datasets. We sample the solution such that each reasoning style (NL vs Code) is distributed equally in the whole fine-tuning dataset. For the evaluation phase, we use questions from the test set of the original GSM8k dataset.

Results. Figure 7 highlights a key point that is easy to miss when considering only overall data diversity: even when the ratio between reasoning modes is the same, varying only how problems are solved leads to significantly different coverage behavior after post-training. Under *problem-level diversity*, performance is stable across epochs, with only a small drop in pass@k at larger k. By contrast, *data-level diversity* exhibits the opposite pattern where pass@1 improves monotonically, but pass@k for larger k initially increases and then degrades sharply as training continues. This effect is most visible at high k, where the reduction in pass@k becomes substantial. Importantly, these differences arise despite both settings having the same overall diversity ratio. The only change is in how that diversity

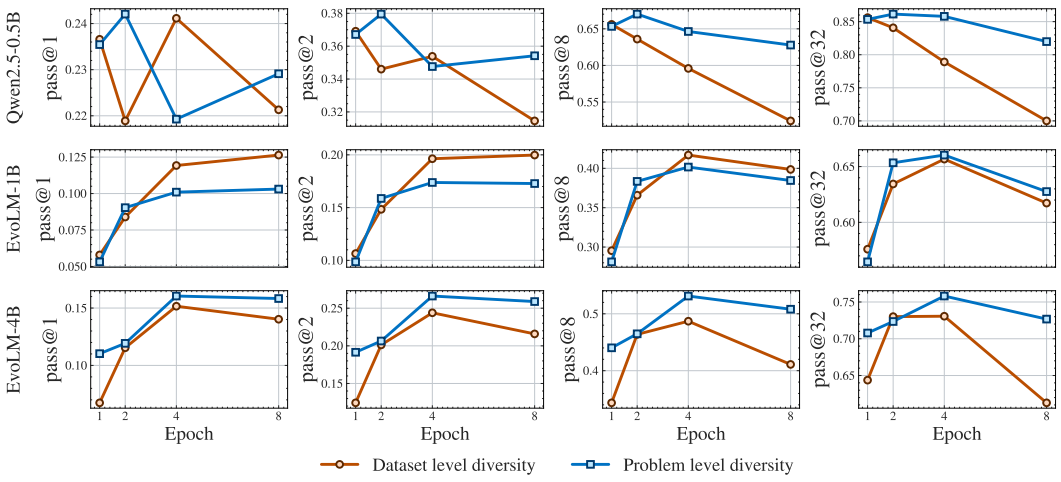
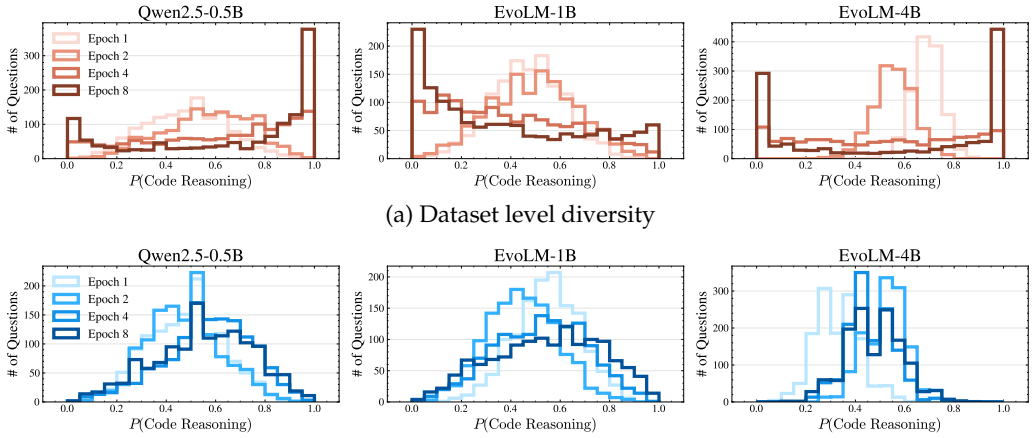


Figure 7: **Effect of data diversity design on coverage in reasoning mode selection task**. Pass@k across SFT epochs for *data-level* vs. *problem-level* data diversity.



(a) Dataset level diversity

(b) Problem level diversity

Figure 8: **Reasoning mode preference under different diversity designs.** Distribution of per problem code vs. NL reasoning across SFT epochs for *data-level* vs. *problem-level* diversity.

is distributed. This suggests that coverage is not determined solely by how much diversity is present; it depends critically on how that diversity is also structured between problems.

To better understand this difference, we also analyze the model's preference over reasoning modes by measuring the probability of generating code vs. NL solutions for each question (as shown in Figure 8). Under *data-level diversity*, the model becomes increasingly confident in selecting a mode per problem, leading to a bimodal distribution that favors either code or NL (**top**). This aligns with the overconfidence at decision points in the graph setting, where increasing certainty concentrates probability mass on a few trajectories, causing coverage shrinkage and a drop in pass@k. In contrast, *problem-level diversity* yields a completely different, more balanced, and calibrated distribution (**bottom**). Rather than committing to a path, it preserves flexibility, which helps maintain coverage across different solution paths.

These observations suggest models must infer which reasoning mode to apply despite an implicit selection mechanism, often driven by annotator preferences or data curation. Mixing styles can thus introduce bias. As we show next, analogous to the fork-in-the-road step, this leads models to rely on spurious features when selecting reasoning modes.

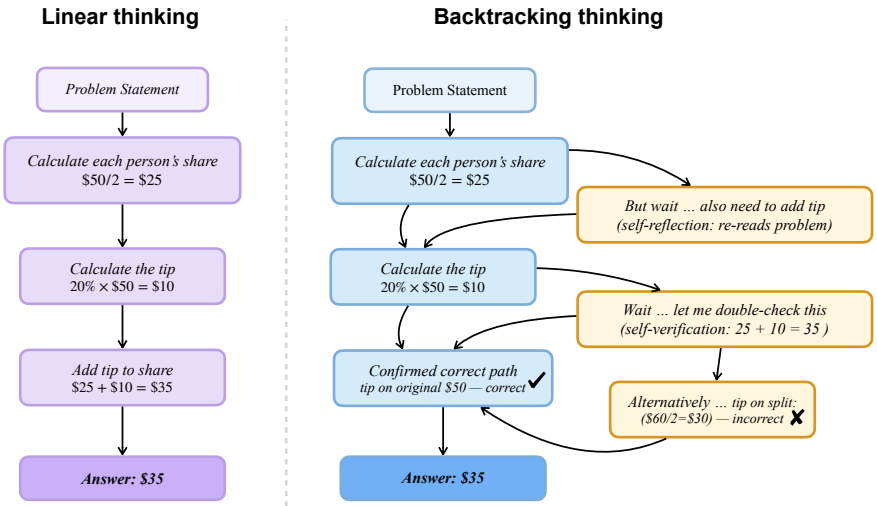


Figure 9: Illustrative examples of linear vs. backtracking reasoning modes.

4.2.2 *Linear vs. Backtracking Reasoning*

Another important mode of variation in reasoning is the *structure* of the reasoning process itself. We focus on two common modes: *linear thinking* and *backtracking thinking*. In linear thinking, the model follows a direct, forward chain of steps without revisiting earlier decisions (**left**). In contrast, backtracking thinking involves self-reflection and verification, where the model revisits intermediate steps (e.g., "wait", "let me check" or "alternatively,") before committing to a solution (**right**). Figure 9 illustrates representative examples of these two reasoning modes.

Setup. Rather than constructing this behavior synthetically through synthetic fine-tuning data, we study it in off-the-shelf R1-distilled reasoning models. According to Guo et al. (2025), these models are trained on mixtures of data generated by both linear and backtracking modes from DeepSeek-V3 and DeepSeek-R1.

As a result, these models naturally exhibit both of these reasoning modes (i.e., showing both instruct and thinking behaviors when needed). We also probe their behavior using CoT-without-prompting decoding mechanism (Wang & Zhou, 2024), and observe these two distinct behaviors in the samples of the same model: (i) linear traces without verification or backtracking, and (ii) backtracking traces with explicit self-reflection patterns. This provides a natural testbed for analyzing reasoning mode selection as a decision point in more general reasoning settings.

Results. An interesting observation is that these models are **highly brittle at the *initial tokens* of the reasoning trace**. Small variations in the first token (e.g., "Okay," vs. "Let" or "To") can lead to drastically different reasoning behaviors out of the models, including changes in structure, length, and accuracy—even for the same input questions. We interpret these **initial tokens as the practical realization of decision points**: they implicitly determine which reasoning mode the model commits to. Similar to indecipherable nodes in the graph branching experiments, the model here also appears to rely on weak or spurious cues (e.g., stylistic prefixes) to select a reasoning path at initial tokens as decision points. In our experiments, we study this through systematically varying prefix tokens and evaluating their performance across multiple benchmark datasets (GSM8K, MATH-500, AIME24, and AIME25). The results in Figure 10 show that small prefix changes can significantly alter both reasoning behavior and performance. In particular, prefixes such as *"Okay"* or *"Alright"* consistently trigger backtracking behavior in thinking traces, leading to considerably longer responses with more verification steps and improved performance on math reasoning benchmarks.

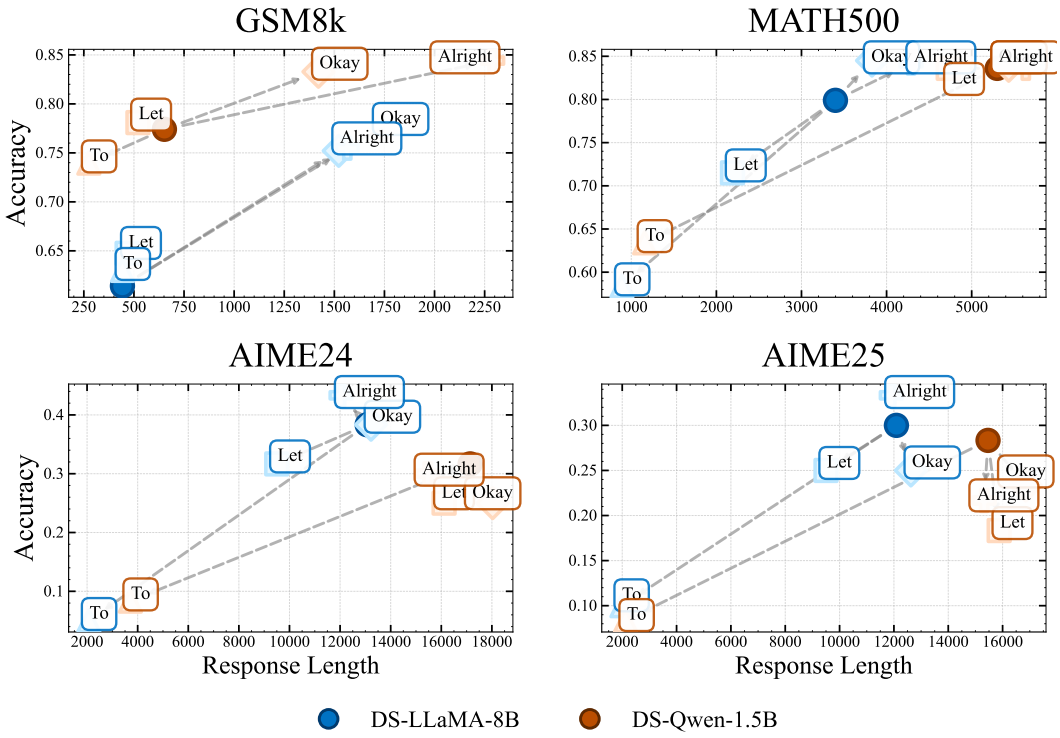


Figure 10: **Impact of prefix token manipulation on reasoning behavior.** Small changes only in one of the initial tokens (e.g., "Okay," "Let," "To") lead to considerably large variations in response length and accuracy across different benchmarks datasets (GSM8K, Math500, AIME 2024/2025).

These results suggest that reasoning structure itself behaves like a fork in the road: the model must commit to either linear or backtracking thinking, and this choice largely determines both the trajectory and final outcome. We find that these decision points are brittle where small, seemingly irrelevant changes in the initial prefix tokens can shift the model between reasoning modes, leading to large differences in behavior and performance. This indicates that reasoning mode selection is driven by weak signals, making it a key source of instability and side effects such as coverage collapse, but also a natural point of intervention for control. In the next section (Sec 5), we leverage this observation and show how manipulating prefix tokens can actually mitigate the coverage shrinkage by encouraging diversity over the reasoning modes.

Over-thinking and Under-thinking. We also perform a fine-grained analysis showing that strategy selection in distilled reasoning models correlates with irrelevant lexical features rather than the problem structure. We evaluate this on (1) simple knowledge questions (e.g., "What is the capital of France?"), where the complex reasoning process is usually unnecessary, as well as (2) counterfactual arithmetic questions (e.g., "What is 6 + 3 in base-7?"), which can be mistaken for standard arithmetic and requires careful reasoning and verification process. A reasonable and efficient approach is to perform

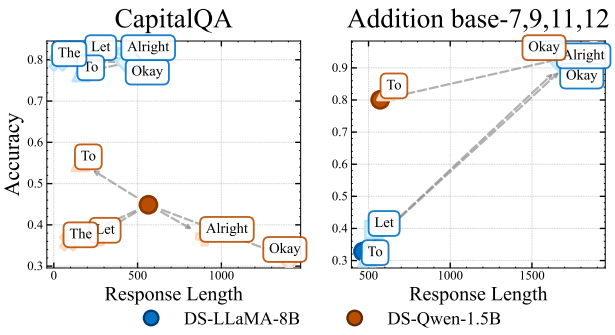


Figure 11: Models' default behavior shows over-thinking on factual QA (backtracking hurts) and under-thinking on counterfactual arithmetic (backtracking helps).

linear, straightforward thinking for the first type of questions and backtracking thinking for the second type of questions. However, our results (presented in Figure 11) show that the models struggle to choose suitable thinking modes in these tasks. Specifically, on counterfactual arithmetic problems, the model's default behavior mostly resembles linear thinking, but enforcing non-linear or backtracking strategies (e.g., through prefixes like "Okay") can boost performance by up to 60%. This shows clear *under-thinking* in models when more reasoning and verification is actually needed. On the other hand, on CapitalQA factual problems, the model performs better with shorter and linear reasoning structure (e.g., through prefixes like "To"), yet its default behavior mostly resembles longer reasoning that hurts performance. This shows *over-thinking* in models when less reasoning is actually sufficient. More detailed examples regarding this observation is provided in Appendix Section B.3.

5 Discussion on Data-inspired Shrinkage Mitigation Strategies

Our analysis demonstrates that coverage shrinkage is a data-driven artifact, where models learn to lock alternative reasoning paths behind spurious decision points. Building on this understanding, we explore simple strategies to mitigate this collapse by directly targeting these decision points.

Data Diversity Design. Our experiments in Section 4.2.1 show that per-problem coverage of alternative decisions can significantly control shrinkage behavior compared to distributing the same level of diversity across the problems. This highlights the importance of post-training data design and distribution of diversity that explicitly accounts for reasoning decision-point structure and its coverage among problems.

First-token Manipulation. Motivated by the observed brittleness of reasoning models to initial prefix tokens, we hypothesize that the **alternative reasoning strategies are still present inside the post-trained models, simply locked behind collapsed decision points**. To test this, we propose a simple inference-time intervention: instead of relying on the default generation, we perturb the initial decision by forcing the model to start from different high-probability initial reasoning tokens. Concretely, for each problem, we uniformly sample from the Top-k candidate prefix tokens (Top-8 in our experiments) and continue solutions. We compare this to standard decoding (Default) and a deterministic Top-1 prefix setting. Figure 12 reports pass@k on GSM8K across SFT checkpoints for `EvoLM-1B` and `Qwen-2.5-0.5B` backbone models finetuned on a MetaMathQA subset (Yu et al., 2023). As observed, all pass@k improve during the early phase of training. However, in later stages, only pass@1 continues to improve while pass@k consistently declines across settings. Our intervention shows that the Top-k prefix setting (Top-8) matches early performance but achieves substantially higher pass@k later, effectively recovering lost coverage in post-trained models. These results show that the brittleness of initial tokens—identified as decision points in Section 4.2.2—can be exploited to control model behavior and recover lost coverage after post-training. **The underlying reasoning capacity is not eliminated but suppressed during training, and can be reactivated via targeted perturbations and diversity at decision points.**

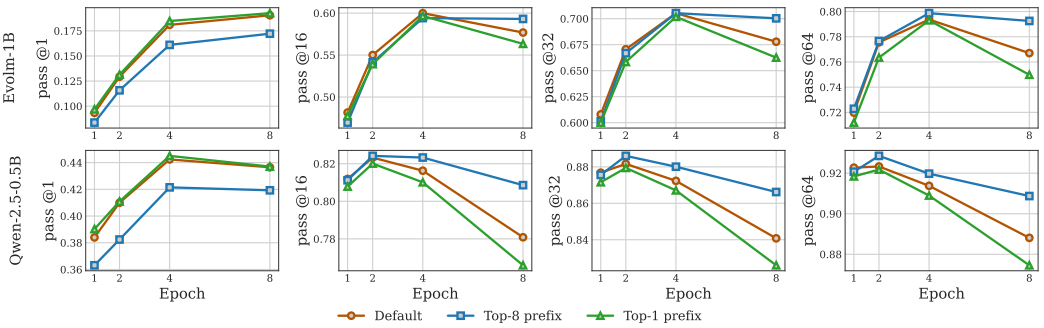


Figure 12: **Recovering coverage via prefix perturbation.** Top-k prefix sampling (Top-8) mitigates coverage shrinkage and improves pass@k at larger k

6 Conclusion

In this work, we investigate the phenomenon of "coverage shrinkage" in post-training reasoning language models. Diverging from prevailing theories that blame the optimization dynamics and post-training algorithms, we propose a data-centric hypothesis and argue that shrinkage is highly driven by "forks in the road" or decision points in fine-tuning data where multiple valid reasoning paths exist, but the rationale for selecting one over the other is obscured. Through experiments on controlled case studies, synthetic tasks, and mathematical reasoning, we demonstrate that models learn to rely on spurious features to resolve the ambiguity in reasoning chains, leading to coverage collapse after the post-training. By mitigating this shrinkage via targeted data diversity and inference-time prefix manipulations, we demonstrate that coverage and reasoning diversity is not lost after post-training, but merely locked behind the uncalibrated decisions. We hope that our insights provide a useful lens for those working on post-training for reasoning, data curation, or test-time scaling.

References

Gregor Bachmann and Vaishnavh Nagarajan. The pitfalls of next-token prediction. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=76zq8Wkl6Z.

Bradley Brown, Jordan Juravsky, Ryan Saul Ehrlich, Ronald Clark, Quoc V Le, Christopher Re, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling, 2025. URL https://openreview.net/forum?id=0xUEBQV54B.

Feng Chen, Allan Raventos, Nan Cheng, Surya Ganguli, and Shaul Druckmann. Rethinking fine-tuning when scaling test-time compute: Limiting confidence improves mathematical reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=jvVQeSMeGM.

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.

Xingyu Dang, Christina Baek, Kaiyue Wen, J Zico Kolter, and Aditi Raghunathan. Weight ensembling improves reasoning in language models. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=S2IKxulLT1.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu,

Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645 (8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Yinghui He, Abhishek Panigrahi, Yong Lin, and Sanjeev Arora. Skill-targeted adaptive training. *arXiv preprint arXiv:2510.10023*, 2025.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024a.

Bowen Jiang, Yangxinyu Xie, Zhuoqun Hao, Xiaomeng Wang, Tanwi Mallick, Weijie J Su, Camillo Jose Taylor, and Dan Roth. A peek into token bias: Large language models are not yet genuine reasoners. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 4722–4756, Miami, Florida, USA, November 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.272. URL https://aclanthology.org/2024.emnlp-main.272/.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=i1uGbfHHpH.

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. *arXiv preprint arXiv:2305.20050*, 2023.

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models. *arXiv preprint arXiv:2505.24864*, 2025.

Seyed Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. GSM-symbolic: Understanding the limitations of mathematical reasoning in large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=AjXkRZIvjB.

Zhenting Qi, Fan Nie, Alexandre Alahi, James Zou, Himabindu Lakkaraju, Yilun Du, Eric Xing, Sham Kakade, and Hanlin Zhang. Evolm: In search of lost language model training dynamics. *arXiv preprint arXiv:2506.16029*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Parshin Shojaee, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity. *arXiv preprint arXiv:2506.06941*, 2025.

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

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data. *arXiv preprint arXiv:2410.01560*, 2024a.

Shubham Toshniwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, and Igor Gitman. Openmathinstruct-1: A 1.8 million math instruction tuning dataset. *arXiv preprint arXiv: Arxiv-2402.10176*, 2024b.

Boxin Wang, Chankyu Lee, Nayeon Lee, Sheng-Chieh Lin, Wenliang Dai, Yang Chen, Yang Chen, Zhuoling Yang, Zihan Liu, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Nemotron-cascade: Scaling cascaded reinforcement learning for general-purpose reasoning models. *ArXiv*, abs/2512.13607, 2025. URL https://api.semanticscholar.org/CorpusID:283897326.

Xuezhi Wang and Denny Zhou. Chain-of-thought reasoning without prompting. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=4Zt7S0B0Jp.

Fang Wu and Yejin Choi. The invisible leash: Why rlvr may not escape its origin. In *2nd AI for Math Workshop@ ICML 2025*, 2025.

Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1819–1862, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.102. URL https://aclanthology.org/2024.naacl-long.102/.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=4OsgYD7em5.

Yi Zhang, Arturs Backurs, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, and Tal Wagner. Unveiling transformers with lego: a synthetic reasoning task. *arXiv preprint arXiv:2206.04301*, 2022.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2024, 2024.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025.

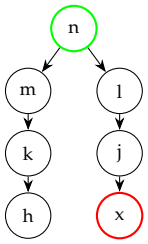
Rosie Zhao, Alexandru Meterez, Sham Kakade, Cengiz Pehlevan, Samy Jelassi, and Eran Malach. Echo chamber: Rl post-training amplifies behaviors learned in pretraining. *arXiv preprint arXiv:2504.07912*, 2025.

A Details on Synthetic Experiment

A.1 Task design

We introduce a synthetic task, designed to capture two core components of mathematical problem solving: (1) chains of variable manipulation and (2) planning over a dependency structure. In this task, the prompt specifies a set of functional dependencies among variables together with a query about the value of a designated target variable. During the SFT phase, the model is trained on solutions consisting of a sequence of intermediate computation steps in order to find the target's value.

Q: Let each letter represent a numerical variable. These variables are defined as follows: $n = 10$; $m = n + 12$; $k = m + 3$; $h = k + 4$; $l = n + 19$; $j = l + 17$; $x = j + 2$. What is the resulting value of x?



A key property of this task is the topology of the underlying dependency graph. In the example shown above, we consider a *path-star* graph, introduced in Bachmann & Nagarajan (2024) to study the failure of the teacher-forcing learning objective. The graph consists of a central root node with multiple outgoing paths. The target variable lies at the end of a unique path. To determine its value, the model must first identify the relevant dependency path and then correctly compute all intermediate variables along that path.

Prompts. We use the Alpaca instruction format. An example of a prompt is shown below:

A problem from the synthetic task

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

Instruction:
Solve the following math problem, and put your final answer within `\boxed{}`.

Input:
Consider a system of variables where each variable is defined as follows: t = x + 10, e = c + 17, s = y + 13, x = a + 2, b = v + 16, j = s + 17, r = d + 18, u = o + 20, h = f + 6, o = v + 6, i = p + 16, y = u + 11, a = h + 5, v = 3, d = z + 13, z = e + 13, g = i + 12, f = m + 11, c = j + 6, p = t + 6, m = b + 3. If $v = 3$, determine the value of g.

Response:

We also provide examples of two variations of the solution in the following.

Example of *Forward* Solution (with decision point)

To find the target value, we compute the following variables step by step:

1. $o = d + 19 = 37$
2. $k = o + 9 = 46$
3. $w = k + 1 = 47$
4. $v = w + 6 = 53$
5. $g = v + 14 = 67$
6. $f = g + 11 = 78$
7. $m = f + 9 = 87$
8. $l = m + 5 = 92$
9. $p = l + 7 = 99$
10. $s = p + 11 = 110$

Thus, $s = \boxed{110}$.

Example of *Reverse* Solution (without decision point)

To find the target value, we compute the following variables step by step:

1. Substitute $p = l + 7$ into the target expression, yielding $s = l + 18$.
2. Substitute $l = m + 5$ into the target expression, yielding $s = m + 23$.
3. Substitute $m = f + 9$ into the target expression, yielding $s = f + 32$.
4. Substitute $f = g + 11$ into the target expression, yielding $s = g + 43$.
5. Substitute $g = v + 14$ into the target expression, yielding $s = v + 57$.
6. Substitute $v = w + 6$ into the target expression, yielding $s = w + 63$.
7. Substitute $w = k + 1$ into the target expression, yielding $s = k + 64$.
8. Substitute $k = o + 9$ into the target expression, yielding $s = o + 73$.
9. Substitute $o = d + 19$ into the target expression, yielding $s = d + 92$.
10. Substitute $d = 18$ into the target expression, yielding $s = 110$.

Thus, $s = \boxed{110}$.

A.2 Experimental settings

Dataset. We generate a list of variables and equations from a star graph with 2 branches, and the lengths from the root to leaf nodes are 10. Next, we choose text templates and generate a pair of questions and ground truth solutions. This process results in a training set of 6400 samples and a testset of 1000 samples.

Training and Evaluation. We supervise finetune the pretrained Qwen-2.5-0.5B model (Team, 2024) and EvoLM-1B (Qi et al., 2025) on the training set for 16 epochs with lr=2e-5. For evaluation, we mark a solution as correct if the boxed output matches the value of the target variable.

A.3 Probing confidence at decision points

We probe the confidence of the model at the decision points of the forward solution by computing the probability of the next token of the following prompt,

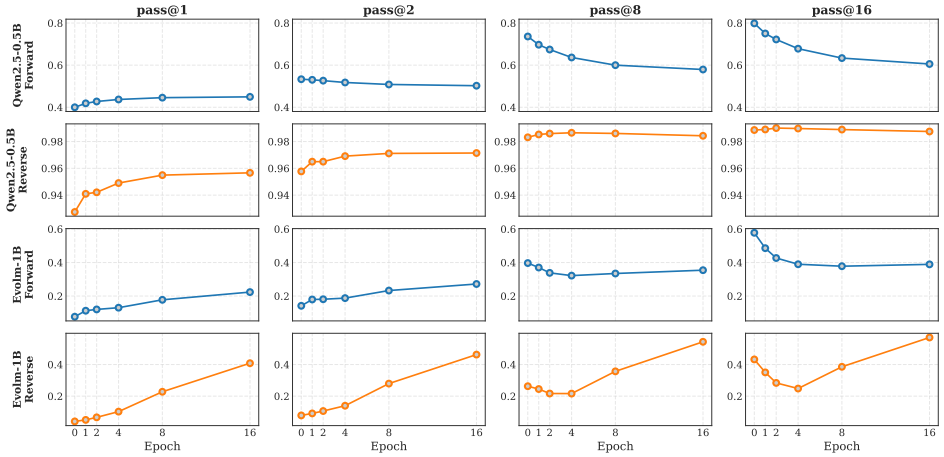


Figure 13: Pass@k performance when running GRPO on models pretrained on forward and reverse (-DP) solutions.

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

Instruction:
Solve the following math problem, and put your final answer within `\boxed{}`.

Input:
Consider a system of variables where each variable is defined as follows: t = x + 10, e = c + 17, s = y + 13, x = a + 2, b = v + 16, j = s + 17, r = d + 18, u = o + 20, h = f + 6, o = v + 6, i = p + 16, y = u + 11, a = h + 5, v = 3, d = z + 13, z = e + 13, g = i + 12, f = m + 11, c = j + 6, p = t + 6, m = b + 3. If $v = 3$, determine the value of g.

Response:
To find the target value, we compute the following variables step by step:

1.

Since the model is trained to predict the name of the variable of the first step after this input, we are able to identify which branch the model will take subsequently and the confidence of following this branch.

A.4 Reinforcement Learning with Verification Reward (RLVR)

Our experiments in the main text focus on SFT dynamics and pass@k across SFT epochs. To examine the role of data and indecipherable decision points under other post-training methods, we apply a standard RLVR algorithm, Group Relative Policy Optimization (GRPO) (Shao et al., 2024), on models that are first supervised fine-tuned for one epoch on either forward (with decision points) or reverse (without decision points) data. The results, shown in Figure 13, demonstrate that the same coverage shrinkage pattern emerges during the RL phase when the training data contains decision points (the Forward setting). These findings shed insight into how pretraining data affects coverage shrinkage during RL; it arises when the optimization is performed over responses that require navigating ambiguous decision points. In contrast, when such blind forks are absent, the RL algorithm preserves coverage.

B Details on Experiment with Reasoning Modes

B.1 Natural Language vs. Code Reasoning

In this experiment, we collect SFT samples from two common mathematical reasoning datasets: OpenMathInstruct-1 (Toshniwal et al., 2024b), and OpenMathInstruct-2 (Toshniwal et al., 2024a). OpenMathInstruct-2 consist of solutions in natural language, while OpenMathInstruct-1 consists of Python code solutions generated by Mixtral-8x7B (Jiang et al., 2024a). We first filter only GSM8k questions and combine them from these datasets. We sample the solution such that each reasoning style (NL vs Code) is distributed equally in the whole dataset. Questions in the test set from GSM8k are used for evaluation.

For the sampling implementation and parameters, we use the default vllm inference server (Kwon et al., 2023) with $temperature = 1.0$, $top\_p = 0.95$, $max\_new\_tokens = 1024$ and generate 64 solutions per test sample. The backbone models we used in this experiment are: `Qwen-2.5-0.5b` (Team, 2024), `EvoLM-1B-160BP-8M42M` and `EvoLM-4B-160BP-8M42M` (Qi et al., 2025).

B.2 Distilled Reasoning Models

Decoding Configuration. In this experiment, we use the `vLLM` library (Kwon et al., 2023) for efficient inference with the following parameters:

- **Temperature:** 0.6
- **Top-p:** 0.95
- **Top-k:** -1
- **Max tokens:** 32768
- **Max model length:** 32768
- **Trust remote code:** `True`

, which are recommended by DeepSeek. To format the input prompt, we use each model's default chat template.

Figure 14 presents the performance of DeepSeek-R1-Distilled reasoning models when generating responses from different thinking prefixes across 4 different reasoning benchmark GSM8k (Cobbe et al., 2021), MATH500 (Lightman et al., 2023), AIME24 (Zhang & Math-AI, 2024), and AIME25 (Zhang & Math-AI, 2025).

B.3 Examples of models' responses

For the counterfactual addition prompt, we use the template from Wu et al. (2024). In the following, we provide examples of questions and responses from models:

Addition in base-12

Question: You are a mathematician. Assuming that all numbers are in base-7 where the digits are "0123456", what is 66 + 50?

Let's think step by step, and end the response with the result in \boxed{}.

Correct answer: 146

Listing 1: Response from DS-LLaMA-8B

```
<think>
First, I need to add the two base-7 numbers 66 and 50.

I'll start by aligning the numbers vertically, ensuring each digit is in the correct
     place value.
```

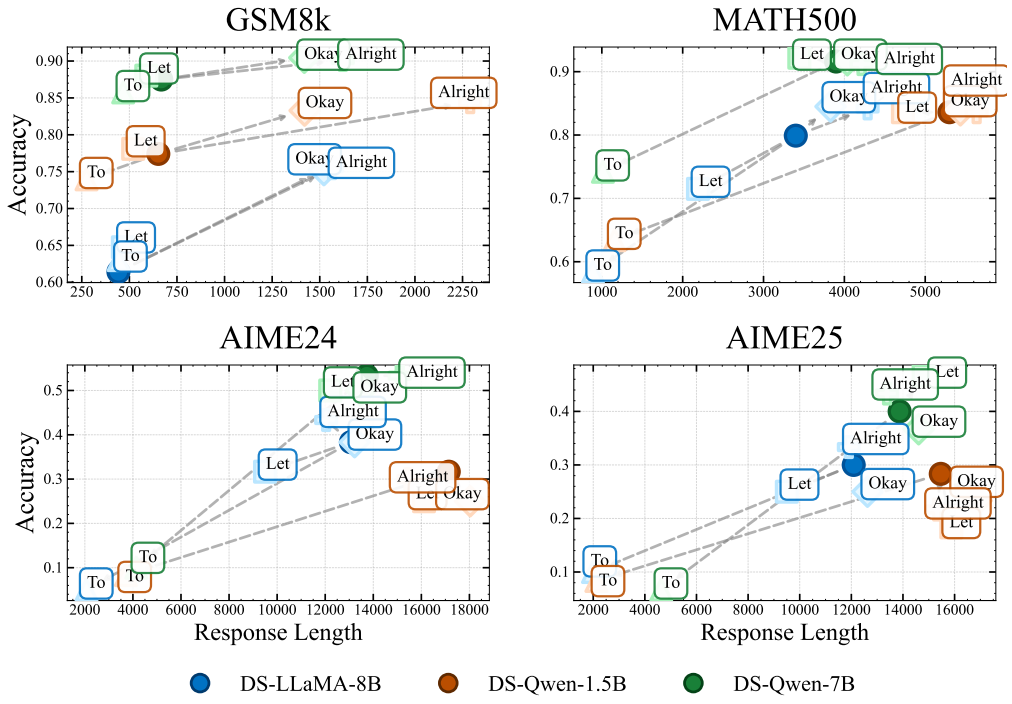


Figure 14: **Performance and Response Length across different prompt prefixes.** The accuracy of two models—DS-Qwen-1.5B, DS-Qwen-7B, and DS-LLaMA-8B—across mathematical benchmarks (GSM8K, MATH500, AIME24, and AIME25). The data demonstrates a high degree of performance variance (up to 20%) and response length variance (up to 5.8x) depending entirely on the starting word or phrase of the thinking prefix.

```
Next, I'll add the digits in each column, starting from the rightmost digit.

In the units place, 6 plus 0 equals 6.

In the sevens place, 6 plus 5 equals 11. Since we're working in base 7, 11 divided
    by 7 is 1 with a remainder of 4. I'll write down the remainder and carry over
    the quotient.

Finally, I'll add the carried over 1 to the next higher place, resulting in a final
    sum of 116 in base 7.
</think>

**Solution:**

To add the two base-7 numbers **66** and **50** in base-7, follow these steps:

1. **Write the numbers vertically, aligning the digits by place value:**

```
  6 6
+ 5 0
--------
```

2. **Add the digits in each column, starting from the right (units place):**

   - **Units place:** 6 + 0 = 6

   - **Sevens place:** 6 + 5 = 11
```

```
3. **Since we're in base-7:**

   - 11 in base-10 is equal to **1** group of **7** (since \(11 \div 7 = 1\) with a
     remainder of 4).

   - Write down the remainder **4** and carry over the quotient **1** to the next
     higher place.

4. **Add the carried over value to the next higher place:**

   - The carried over **1** is added to the next (sevens) place: 1 (from the carry)
     + 1 (from the carry) = **2**

5. **Write down the final result:**

   Combining all the results, we get **116** in base-7.

**Final Answer:**

\[
\boxed{116}
\]
```

Listing 2: Response from DS-LLaMA-8B with "Okay" prefix

```
<think>
Okay, so I have this problem here: I need to add two numbers, 66 and 50, but both
     numbers are in base-7. Hmm, base-7, that's not our usual base-10 system we use
     every day. I remember that in base-7, each digit represents a power of 7, just
     like in base-10 each digit is a power of 10. So, to add these numbers, I guess
     I need to first convert them into base-10, add them, and then convert back to
     base-7. Or maybe I can add them directly in base-7 without converting. I'm not
     sure which is easier. Let me think.

First, let me recall how base-7 addition works. In base-7, each digit can only be
     from 0 to 6. So, if I add two digits and the sum is 7 or more, I have to carry
     over to the next higher place value. For example, if I'm adding 5 + 6 in base-7,
      that would be 11 in base-10, which is 1*7 + 4, so I would write down 4 and
     carry over 1. Okay, got it.

So, let's write down the numbers vertically, aligning the digits by their place
     values. Let me write 66 and 50 in base-7:

```
  6 6
+ 5 0
--------
```

Wait, hold on. Let me clarify the place values. In base-7, the rightmost digit is
     the units place (7^0), and the next is the sevens place (7^1), then 49s (7^2),
     and so on. So, 66 in base-7 is 6*7^1 + 6*7^0, which is 42 + 6 = 48 in base-10.
     Similarly, 50 in base-7 is 5*7^1 + 0*7^0, which is 35 + 0 = 35 in base-10. So,
     adding them together, 48 + 35 = 83 in base-10. Then, converting 83 back to base
     -7: 7^2 is 49, so 83 divided by 49 is 1 with a remainder of 34. 34 divided by 7
      is 4 with a remainder of 6. So, 83 in base-7 is 1*49 + 4*7 + 6 = 1 4 6, so 146
      in base-7.

Wait, but maybe I should try adding them directly in base-7 without converting to
     base-10. Let me try that. So, writing them vertically:

```
  6 6
```
```

```
+ 5 0
--------
```

Starting from the rightmost digit: 6 + 0. That's easy, 6. Then, moving to the left, 6 + 5. Hmm, 6 + 5 is 11 in base-10. But in base-7, 11 divided by 7 is 1 with a remainder of 4. So, I write down 4 and carry over 1. There are no more digits to add, so I just add the carried-over 1 in front. So, the result is 1 4 6, which is 146 in base-7. That matches my previous result. Okay, that's reassuring.

Wait, let me make sure I didn't make any mistakes. Let me check each step again. Starting from the right: 6 + 0 is 6, no carry-over. Then, 6 + 5 is 11 in base-10. Since we're in base-7, 11 divided by 7 is 1 with a remainder of 4. So, we write down 4 and carry over 1. Since there are no more digits, we just add the carried-over 1 at the beginning. So, yes, 146 in base-7. Converting back to base-10, 1*49 + 4*7 + 6*1 = 49 + 28 + 6 = 83. Which is correct because 66 in base-7 is 48 and 50 in base-7 is 35, 48 + 35 = 83. So, both methods give the same result.

I think I got it right. So, the answer is 146 in base-7, which is 83 in base-10. Therefore, the final answer is 146 base-7.
</think>

To solve the addition of 66 and 50 in base-7, we can perform the addition directly in base-7:

```
  6 6
+ 5 0
--------
```

1. Add the rightmost digits: 6 + 0 = 6 (no carry-over).
2. Add the next digits: 6 + 5 = 11 in base-10. Since we're in base-7, 11 divided by 7 is 1 with a remainder of 4. Write down 4 and carry over 1.
3. Add the carried-over 1 to the front.

Result: 1 4 6 in base-7.

So, the sum of 66 and 50 in base-7 is \boxed{146}.

Example of CapitalQA

Question: What is the capital of Greece?
Correct answer: Athens

Listing 3: Response from DS-Qwen-1.5B with "The" prefix

```
<think>
  The capital of Greece is Athens. It is the most populous city in the country and the capital of both the Peloponnese and the island of Greece.
</think>

The capital of Greece is Athens.
```

Listing 4: Response from DS-Qwen-1.5B with "Okay" prefix

```
<think>
Okay, so I need to figure out the capital of Greece. Hmm, I remember that Greece is a country in Europe, but I'm not exactly sure where its capital is. I think it's a major city, maybe in a coastal area because of the sea. I've heard of
```

```
    places like Athens and maybe Thessaloniki, but I'm not certain. I should
    probably break this down step by step.

First, I'll start by recalling any information I know. I know that Greece is often
    referred to as the ''Venice of the Mediterranean.'' I think that's a nickname,
    but I'm not sure if that helps. I also remember that Athens is the capital of
    Greece. Wait, but I think there's another city called Thessaloniki. Maybe that'
    s another capital? Or perhaps I'm mixing up capitals with something else.

I should consider the geography. Greece has a lot of islands, including the
    Peloponnese, which runs from Sparta to Athens. So, Athens is a major city there.
     I think Thessaloniki is located in the southern part of the country, maybe in
    the Ionian Sea. But I'm not sure if it's the capital.

I think the capital is actually Thessaloniki. But I should verify this. I remember
    reading somewhere that the capital is called the ''City of the North'' because
    of its location in the north of Greece. But I'm not entirely sure. Maybe I can
    think of other cities in Greece and see which one is more likely to be the
    capital.

I know that Athens is the capital of Athens, which is a major city in Greece, and it
    's the capital of another country, but that's not relevant here. Maybe
    Thessaloniki is more of a regional capital or a city-state. I think in some
    contexts, especially in discussions about Greece, Thessaloniki is considered
    the capital, but I'm not certain.

I should also consider the population and the size of the city. I believe
    Thessaloniki has a significant population, and it's a major city in the south
    of Greece. Athens is a much larger city, but it's also a major city. So, which
    one is the actual capital?

I think I've heard conflicting information before. Sometimes, people refer to Athens
     as the capital, but others say it's Thessaloniki. I should try to find a
    reliable source or think about the usual way capitals are named. In many
    European countries, the capital is named after the capital of the country, but
    I'm not sure if that applies here.

Wait, I think in the case of Greece, the capital is actually Thessaloniki. It's the
    capital of the island of Thessalonica, which is part of Greece. So, maybe the
    capital is called Thessaloniki, but it's the capital of the island. That makes
    sense because the island is in the south of the country, so the capital would
    be in the south.

Alternatively, I could think about the layout of Greece. The capital is in the south
    , near the mainland, so it's more likely to be a regional capital. But I'm not
    entirely sure. I think the capital is actually Thessaloniki, so I should go
    with that.
</think>

The capital of Greece is Thessaloniki.
```