The Impact of Automated Reinforcement Learning Fine-Tuning on Advanced Prompt Engineering for Code-Based Reasoning

Guilherme Louro de Salignac e Souza[1], **Marco Cristo**[1], **Eulanda Miranda dos Santos**[1]

[1]Instituto de Computação (IComp) – Universidade Federal do Amazonas (UFAM)
Av. Gen. Rodrigo Octávio 6200, Coroado I, Manaus – AM, 69080-900

{guilherme.louro, marco.cristo, emsantos}@icomp.ufam.edu.br

Abstract. *Prompt Engineering (PE) techniques such as Program of Thoughts (PoT) and Structured Chain-of-Thought (SCoT) were originally proposed to induce structured reasoning in Large Language Models (LLMs). However, the emergence of Large Reasoning Models (LRMs), which internalize reasoning through Reinforcement Learning, raises questions about whether such techniques remain useful. This study compares Zero-shot, Chain-of-Thought (CoT), PoT, and SCoT on an LLM, LLaMA-3.1-8B-Instruct, and its corresponding LRM, DeepSeek-R1-Distill-LLaMA-8B, using the GSM8K dataset. DeepSeek significantly outperformed LLaMA across all prompt strategies, with gains ranging from +26.7 to +43.7 p.p. Within LLaMA, structured prompting mainly induced reasoning, with SCoT achieving the best performance. Within DeepSeek, CoT did not improve over Zero-shot, while PoT and SCoT achieved statistically similar accuracy despite the higher inferential cost of SCoT. These results suggest that PE remains relevant for LRMs, but its role shifts from eliciting reasoning to modulating its structure and cost.*

1. Introduction

Large Language Models (LLMs) have become one of the main technologies in contemporary Artificial Intelligence, driving advances in tasks such as text generation, programming, question answering, and mathematical problem solving. In particular, tasks that require logical reasoning and multi-step planning remain challenging, as they demand not only linguistic fluency but also the ability to decompose problem statements, organize premises, execute intermediate operations, and maintain consistency until the final answer. Despite their strong performance in several scenarios, LLMs are still susceptible to logical inconsistencies, accumulated errors throughout the solution process, and high sensitivity to prompt formulation [Jiang et al. 2024, Shi et al. 2023]. In response to these limitations, the research community has increasingly employed Prompt Engineering (PE) techniques to induce more analytical behaviors and structure the models' inference process.

In this context, PE techniques such as Chain-of-Thought (CoT), Program of Thoughts (PoT), and Structured Chain-of-Thought (SCoT) have been proposed to guide the inferential process through strategies such as task decomposition, demonstrative examples, and explicit instructions, compensating for architectural and training limitations of earlier models [Sahoo et al.]. CoT encourages the model to produce a sequence of intermediate steps before the final answer, favoring problem decomposition in natural

language [Wei et al. 2022]. Code-based approaches, such as PoT and SCoT, seek to go beyond textual explanation, since PoT transfers part of the reasoning process to an executable representation, whereas SCoT organizes intermediate reasoning through structures close to programming primitives [Chen et al. 2023, Li et al. 2025]. These techniques were originally developed to compensate for limitations of traditional LLMs, acting as external mechanisms for inducing reasoning.

The emergence of Large Reasoning Models (LRMs), such as DeepSeek-R1, changes this scenario. Unlike LLMs, LRMs are trained to produce long and explicit chains of reasoning before the final answer, making reasoning a capability embedded in the inferential process itself [Guo et al. 2025, Marjanovic et al. 2026]. Indeed, Guo et al. recommend against using explicit PE techniques such as CoT. This shift raises the central question of whether, in models that already reason spontaneously, classical PE techniques remain useful or become redundant. This work builds on the hypothesis that PE does not disappear with LRMs, but rather changes its function. In LLMs, techniques such as CoT, PoT, and SCoT tend to induce reasoning; in LRMs, they modulate the structure, cost, and formalization of a reasoning process that already emerges naturally. Thus, the problem is no longer only to make the model reason, but also to control when, how much, and how this reasoning is produced.

This work investigates this hypothesis through a comparison between LLaMA-3.1-8B-Instruct, used as a representative LLM, and DeepSeek-R1-Distill-LLaMA-8B, used as a representative LRM, on a sample from the GSM8K benchmark. Four PE techniques are evaluated: Zero-shot, CoT, PoT, and SCoT. The quantitative analysis considers accuracy and paired McNemar tests, while the reasoning chains produced by DeepSeek are analyzed according to the taxonomy introduced by *DeepSeek-R1 Thoughtology* [Marjanovic et al. 2026]. The results show that DeepSeek significantly outperforms LLaMA across all prompt strategies. However, PE plays different roles in each paradigm: in LLaMA, structured prompts mainly induce reasoning, whereas in DeepSeek they modulate the structure and cost of reasoning. In particular, CoT does not significantly improve over Zero-shot in DeepSeek, while PoT and SCoT achieve statistically similar accuracy despite the substantially higher inferential cost of SCoT. These findings suggest that PE remains relevant for LRMs, but its function shifts from eliciting reasoning to controlling how reasoning is organized and formalized.

2. Related Work

Historically, the research community has relied on PE to induce and guide reasoning capabilities that were not intrinsic to LLMs. Among the various techniques, Chain-of-Thought (CoT) has stood out by demonstrating that instructions capable of encouraging the model to make intermediate reasoning steps explicit before the final answer can substantially improve performance in tasks requiring multi-step inference, such as arithmetic, logical, and symbolic problems [Wei et al. 2022]. For tasks that demand logical precision and code synthesis, techniques that use code not only as the final product but also as a structured medium for the inferential process itself are particularly relevant, an approach often referred to as *code-enhanced reasoning* [Yang et al.].

In this context, extensions of CoT have been proposed to make reasoning more precise and, in some cases, executable. Program of Thoughts (PoT), introduced by

[Chen et al. 2023], proposes that the model generate a program, for example in Python, capable of representing the logic of the problem and delegating the execution of calculations to an external interpreter. In parallel, Structured Chain-of-Thought (SCoT), proposed by [Li et al. 2025], seeks to organize intermediate reasoning through programming primitives, such as sequence, branching, and iteration, before generating the final code. Together, these techniques aim to overcome limitations of traditional LLMs in tasks that combine logical reasoning, calculation, and code generation [Sahoo et al.].

A recent paradigmatic advance is the emergence of Large Reasoning Models (LRMs). Unlike LLMs, which require external prompting stimuli to reason, LRMs have internalized the generation of long chains of explicit reasoning, or "thoughts", as a default behavior, resulting from training optimized through Reinforcement Learning (RL) [Xu et al. 2025]. Models such as OpenAI o1 [OpenAI 2024] and, in an open and accessible form, DeepSeek-R1, exemplify this shift. The latter, in particular, demonstrated that robust reasoning capabilities can emerge mainly from large-scale RL, without a mandatory preliminary supervised fine-tuning phase, and that these capabilities can be distilled into smaller models, making research with LRMs more feasible [Guo et al. 2025].

This advance redefines the role of PE. Techniques designed to elicit reasoning, such as standard CoT, potentially become redundant when applied to a model that already reasons spontaneously [Xu et al. 2025]. Recent literature explores how to interact with LRMs, either through inference-time scaling, encouraging the model to "think more" during inference, or through prompts that seek to guide or format its native thought flow [Muennighoff et al. 2025]. However, an open question, and the core of this research, is whether specialized code-based PE techniques, such as PoT and SCoT, whose value has been demonstrated in traditional LLMs, still provide a measurable benefit when applied to LRMs such as DeepSeek-R1.

The evaluation of the quality of model reasoning has also motivated studies that go beyond the analysis of the final answer. Rather than considering only the output, recent works argue that the inferential behavior of models must be examined in order to better understand their limitations, error patterns, and degree of reliability [Mondorf and Plank 2024]. With the advent of LRMs, this analysis has become more direct, since models such as DeepSeek-R1 provide extensive chains of thought before the final answer. In this context, [Marjanovic et al. 2026] analyzed the thoughts generated by DeepSeek-R1 and identified recurring patterns in its reasoning process, including problem definition stages, initial decomposition, reconstruction cycles, and final decision. This work is positioned at the intersection of PE, LRMs, and chain-of-thought analysis, specifically investigating how structured code-based prompts, such as PoT and SCoT, affect not only final performance, but also the organization of the inferential process in a reasoning model.

Unlike prior work that primarily evaluates prompting techniques on conventional LLMs or analyzes reasoning traces of LRMs in isolation, this study compares the same PE strategies across an instruction-tuned LLM and a reasoning-oriented distilled model of similar scale. The contribution is therefore not a new prompting method, but an empirical analysis of how the role of PE changes when reasoning is already internalized by the model.

3. Methodology

3.1. Dataset

To compare the methods, the GSM8K dataset, Grade School Math 8K, was used [Cobbe et al. 2021]. This benchmark was selected because it represents a multi-step arithmetic reasoning challenge in natural language and is widely recognized and used in the literature to evaluate the ability of models to decompose and solve complex mathematical problems [Wei et al. 2022]. Composed of 7473 training examples and 1319 test examples, each dataset instance consists of a textual problem and a step-by-step solution that leads to a final numerical answer. The structure of the problems requires models to perform inference, planning, and execution of a sequence of basic arithmetic operations, including addition, subtraction, multiplication, and division, to arrive at the correct answer.

3.2. Selected Models

The comparative analysis in this study is based on a pair of models with equivalent architecture and scale, both with 8 billion parameters, but trained under different protocols: LLaMA-3.1-8B-Instruct[1], subjected to supervised fine-tuning, and DeepSeek-R1-Distill-LLaMA-8B[2], aligned through Reinforcement Learning. This configuration provides a controlled comparison between two models of similar scale and related architecture, but it does not isolate the causal effect of RL alone. The comparison should therefore be interpreted as an empirical contrast between the selected instruction-tuned LLM and the selected reasoning-oriented distilled model.

3.3. Prompt Engineering Techniques

Four PE techniques were evaluated: (i) Zero-shot, which corresponds to our experimental baseline. The model receives only the problem statement to answer. (ii) Chain-of-Thought (CoT), which instructs the model to present a step-by-step solution before providing the final answer, inducing the decomposition of the problem into intermediate steps in natural language. (iii) Program of Thoughts (PoT), which guides the model to solve the problem through code, structuring the solution as a Python function. (iv) Structured Chain-of-Thought (SCoT), which induces the model to build a plan based on programming primitives. Then, the model produces a Python implementation of the solution based on the plan.

To support reproducibility, the exact prompt templates used for all strategies will be released in an accompanying GitHub[3] repository.

3.4. Reasoning Chain Taxonomy

Following [Marjanovic et al. 2026], this taxonomy organizes reasoning into four stages. DEFINE corresponds to problem reformulation, identification of relevant information, and clarification of the target answer. BLOOM represents the first main reasoning cycle, in which the model decomposes the problem, organizes premises, and constructs an initial solution. CYCLE captures later reconsideration, verification, or reconstruction steps, including revisions of previous hypotheses or consistency checks. FINAL marks the consolidation of the reasoning process and the presentation of the final answer.

[1] https://huggingface.co/meta-LLaMA/LLaMA-3.1-8B-Instruct

[2] https://huggingface.co/DeepSeek-ai/DeepSeek-R1-Distill-LLaMA-8B

[3] https://github.com/guilherme-louro/pe-vs-lrm

3.5. Experimental Protocol

The experimental protocol was divided into three main stages: answer generation, final-answer extraction, and reasoning-chain analysis. All reported evaluations were conducted on a random sample of 247 examples from the GSM8K test set, selected with a fixed seed due to computational cost. The same sampled instances were used for all models and prompt strategies, ensuring paired comparisons. In all experiments, temperature $T = 0.6$ and $top\text{-}p = 0.95$ were used, following the recommendations adopted in the DeepSeek-R1 tests [Guo et al. 2025]. For reproducibility, we used a single generation per model, prompt, and instance, with fixed generation parameters across all conditions. All model generations were performed in Google Colab using an NVIDIA L4 GPU and the Hugging Face Transformers library. Each problem was submitted to LLaMA-3.1-8B-Instruct and DeepSeek-R1-Distill-LLaMA-8B under the four PE techniques: Zero-shot, CoT, PoT, and SCoT, totaling 1,976 model queries.

After answer generation, correctness was assessed from the final numerical answer associated with each model output. For the Zero-shot and CoT settings, the final answer was extracted using a rule-based procedure that searched for the last explicit numerical answer in the output and normalized formatting differences such as commas, currency symbols, and trailing punctuation. For the PoT and SCoT settings, the generated code was extracted using a rule-based script and executed in an external Python interpreter to obtain the corresponding numerical answer. When the script failed to automatically extract the code, the code block was manually inspected and extracted whenever possible. However, some outputs did not follow the expected PoT or SCoT format and did not provide executable code. In such cases, the final textual answer generated by the model was used for evaluation, following the same rule-based extraction procedure adopted for Zero-shot and CoT. Ambiguous cases were manually inspected before being classified as correct or incorrect. These binary correctness labels formed the basis for accuracy calculation and for the paired matrices used in the McNemar tests.

Next, the reasoning chains produced by DeepSeek were preserved and analyzed separately. Since this model provides explicit reasoning segments before the final answer, the chains were segmented according to the taxonomy introduced by *DeepSeek-R1 Thoughtology*. To avoid relying blindly on automatic annotation, the authors first conducted a manual audit of 30 randomly selected chains, balanced across the four PE strategies and answer correctness. These chains were used to validate the interpretation of the DEFINE, BLOOM, CYCLE, and FINAL categories and to provide examples for the annotation prompt. Inter-annotator agreement reached a Cohen's Kappa of approximately 0.78. The remaining chains were then annotated with the OpenAI ChatGPT 5.5, accessed through the ChatGPT interface in May to June 2026, using the original Thoughtology prompt enriched with the manually validated examples.

4. Results and Discussion

4.1. Quantitative Comparison between LLaMA and DeepSeek

Table 1 summarizes the quantitative comparison between the evaluated models and prompt techniques. The left block reports the accuracy obtained by LLaMA-3.1-8B-Instruct (LLM) and DeepSeek-R1-Distill-LLaMA-8B (LRM), while the right block reports the internal pairwise comparisons among prompt techniques. In the accuracy-gain

block, values marked with * indicate statistically significant LLM–LRM differences according to the paired McNemar test ($p < 0.001$). In the pairwise block, each cell follows the format LRM, LLM; $>$, $<$, and $\sim$ indicate, respectively, significantly higher accuracy, significantly lower accuracy, and no statistically significant difference for the row technique compared to the column technique. For example, the cell at row PoT and column CoT is $>, \sim$, meaning that PoT significantly outperformed CoT for the LRM, while the difference between PoT and CoT was not statistically significant for the LLM.

Table 1. Accuracy gains and pairwise comparisons among prompt techniques.

Accuracy and gain				Pairwise comparisons				
Tech.	**LLM**	**LRM**	**Gain (p.p.)**	**Tech.**	**Zero-shot**	**CoT**	**PoT**	**SCoT**
Zero-shot	33.2	76.9	+43.7*	**Zero-shot**	–	$\sim, <$	$<, <$	$<, <$
CoT	46.2	74.5	+28.3*	**CoT**	$\sim, >$	–	$<, \sim$	$<, <$
PoT	42.5	85.8	+43.3*	**PoT**	$>, >$	$>, \sim$	–	$\sim, <$
SCoT	59.5	86.2	+26.7*	**SCoT**	$>, >$	$>, >$	$\sim, >$	–

As shown in Table 1, DeepSeek achieved higher accuracy than LLaMA across all evaluated prompt strategies. The gains ranged from +26.7 percentage points in SCoT to +43.7 percentage points in Zero-shot. The largest gains were observed in Zero-shot and PoT, where DeepSeek exceeded LLaMA by +43.7 and +43.3 percentage points, respectively. These results indicate that the superiority of the reasoning model is not restricted to a single prompting strategy, but occurs consistently across Zero-shot, CoT, PoT, and SCoT.

The comparison also shows that structured prompting partially reduces, but does not eliminate, the gap between the models. In SCoT, LLaMA reached its highest accuracy, 59.5%, while DeepSeek achieved 86.2%. This suggests that SCoT provides useful external structure for the LLM, but the LRM remains substantially more robust in multi-step mathematical reasoning. Overall, these results support the hypothesis that LRMs exhibit greater ability to solve problems that require decomposition, planning, and the execution of intermediate operations.

4.2. Effect of PE Techniques within Each Model

The pairwise-comparison block in Table 1 shows that the effect of PE techniques differs between the two model paradigms. For LLaMA, SCoT was the strongest technique, significantly outperforming Zero-shot, CoT, and PoT. CoT and PoT also improved over Zero-shot, although CoT and PoT were not significantly different from each other. This pattern suggests that, in the LLM, prompt techniques operate mainly as mechanisms for inducing reasoning by providing external structure for problem decomposition and solution construction.

For DeepSeek, the pattern was different. CoT did not significantly outperform Zero-shot, suggesting that a generic instruction to reason step by step may be partially redundant in an LRM, since the model already tends to produce structured reasoning as part of its inferential behavior. In contrast, PoT and SCoT significantly outperformed both Zero-shot and CoT, indicating that code-based prompting can still influence the reasoning process of the LRM. However, PoT and SCoT were statistically indistinguishable from each other.

The comparison between PoT and SCoT is particularly important. For LLaMA, SCoT significantly outperformed PoT, suggesting that the explicit planning structure introduced by SCoT helped the LLM organize the solution more effectively. For DeepSeek, however, the additional structure introduced by SCoT did not translate into a statistically significant accuracy gain over PoT. This result indicates that, in LRMs, PE techniques no longer act only as triggers for reasoning. Instead, they appear to modulate the form, cost, and organization of a reasoning process that already emerges naturally. To investigate this modulation, the following subsections analyze the reasoning chains produced by DeepSeek.

4.3. Inferential Cost of PE Techniques

Figure 1a presents the average number of tokens per technique, with a 95% confidence interval. Zero-shot and CoT exhibited the lowest average costs, with approximately 253 and 240 tokens, respectively. PoT showed an intermediate cost, with an average of 459 tokens. SCoT, in turn, presented approximately 460 tokens in the code-generation phase analyzed by the reasoning-chain taxonomy. However, when the additional planning phase is considered, which averaged 477 tokens, its total inferential cost increases to approximately 940 tokens, making it the most expensive technique. This increase in SCoT is related to the nature of the technique itself. Unlike PoT, which tends to directly produce a solution in the form of a program or computational reasoning, SCoT includes an additional structured planning stage before the final answer.

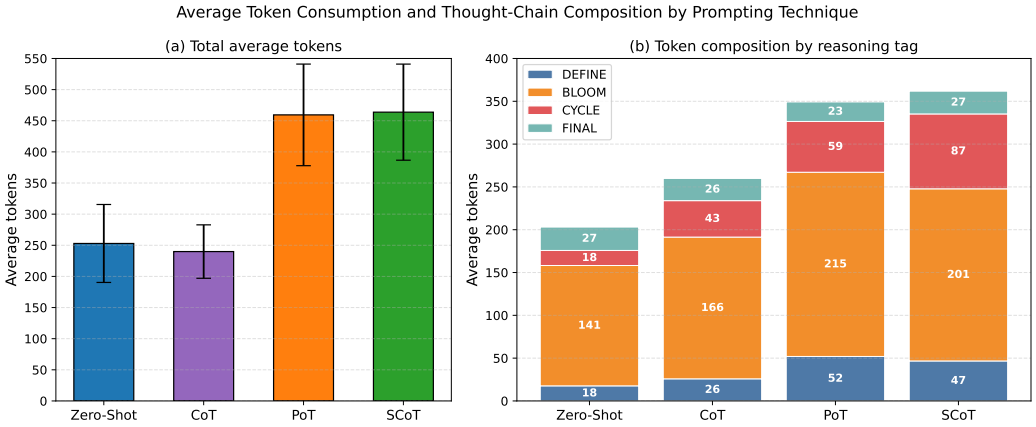


Figure 1. Average token count by prompting technique, excluding SCoT planning tokens. (a) Total average number of tokens. (b) Token composition by reasoning tag.

The comparison between PoT and SCoT is particularly relevant. Since Table 1 shows that PoT and SCoT do not differ significantly in accuracy for DeepSeek, the cost difference becomes an important analysis criterion. Although SCoT may produce a more formalized and auditable chain, this structural gain did not translate into a statistically significant advantage over PoT. Thus, the results indicate that more structured techniques may increase inferential cost without generating a proportional performance gain.

This finding reinforces that the evaluation of LRMs should not be limited to accuracy. Since these models can produce extensive reasoning chains, it is also necessary

to consider efficiency, understood as the relationship between performance and computational cost. In this sense, PoT emerges as a more efficient alternative than SCoT in the evaluated set, as it achieves statistically equivalent performance with a lower average number of tokens.

4.4. Composition of Reasoning Chains

Figure 1b presents the absolute composition of reasoning chains by technique, considering the *DEFINE*, *BLOOM*, *CYCLE*, and *FINAL* stages. In the case of SCoT, we did not include the tokens of the additional planning phase (477 tokens) since they were not manually inspected. Across all strategies, the *BLOOM* stage concentrated most of the tokens, indicating that the initial decomposition of the problem represents the core of the reasoning produced by the model.

In Zero-shot, the average number of tokens in *BLOOM* was 141, while the remaining stages showed lower values: 18 tokens in *DEFINE*, 18 in *CYCLE*, and 27 in *FINAL*. In CoT, a general increase in the chain was observed, with 166 tokens in *BLOOM* and 43 in *CYCLE*. In PoT, the structure became even more extensive, with 215 tokens in *BLOOM*, 59 in *CYCLE*, and 52 in *DEFINE*. In SCoT, in addition to the usual chain stages, a planning phase with an average of 477 tokens was observed, accounting for a large part of the total cost of the technique.

These results show that PE techniques alter not only the total number of tokens, but also the distribution of inferential effort across different reasoning phases. The predominance of *BLOOM* suggests that the first decomposition of the problem is the most important stage of the process. On the other hand, the increase in *CYCLE* in techniques such as PoT and SCoT indicates that more structured prompts stimulate more verification, reconstruction, or reconsideration of hypotheses.

However, the greater presence of *CYCLE* should not automatically be interpreted as an improvement in reasoning. Reconstruction cycles may act as productive verification mechanisms, but they may also reflect uncertainty or unproductive repetition. For this reason, the following subsection analyzes whether the number of cycles is associated with correct or incorrect answers.

4.5. Reconstruction Cycles and Relationship with Correctness

Figure 2 presents the distribution of the number of cycles by technique, separating correct and incorrect answers. Across all strategies, most answers were concentrated in chains without cycles. This pattern is particularly evident in Zero-shot and CoT, in which responses with zero cycles predominate among both correct and incorrect answers.

In PoT and SCoT, there is a greater presence of chains with one cycle, especially among correct answers. However, incorrect answers also contain cycles, and the presence of one or more cycles does not clearly separate correct from incorrect responses. In addition, answers with two or more cycles are less frequent, but appear in all techniques, indicating that additional cycles occur both in successful cases and in failure cases.

This result suggests that the number of cycles, by itself, is not a good indicator of answer quality. The existence of *CYCLE* may indicate that the model revises a hypothesis, verifies an interpretation, or attempts to correct a previous solution. However, it may also

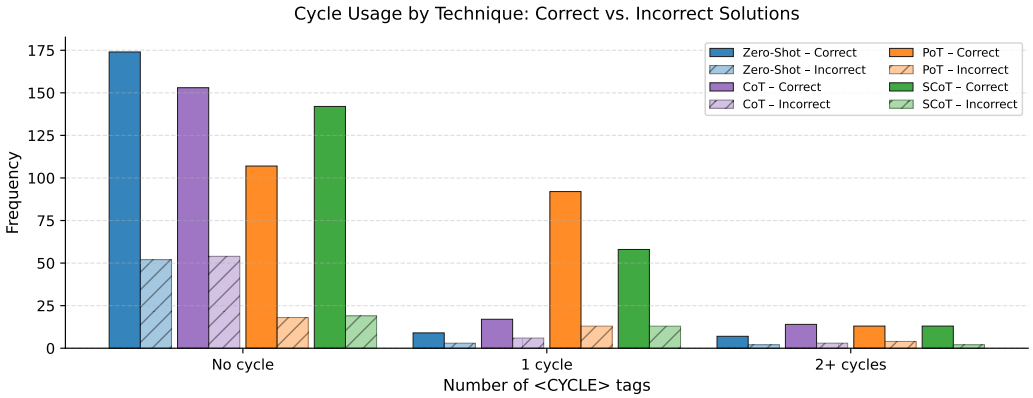


Figure 2. Number of cycles by technique and correctness.

indicate hesitation, uncertainty, or rumination. Thus, the effectiveness of a cycle depends on its semantic quality, not merely on its occurrence.

This interpretation is important for the analysis of LRMs. Since these models are capable of producing long chains of thought, it would be tempting to assume that more reconstructions imply more robust reasoning. The results indicate that this relationship is more complex. Additional reasoning may be useful when it corrects an inadequate initial decomposition, but it may be unproductive when it only prolongs an incorrect trajectory.

4.6. Costly Errors and Chain Length

Figure 3 compares the average number of tokens between correct and incorrect answers for each technique. In Zero-shot and CoT, the average number of tokens for correct and incorrect answers is relatively similar, with a slight reduction in errors. In PoT and SCoT, however, a different pattern is observed: incorrect answers present higher average token counts than correct answers.

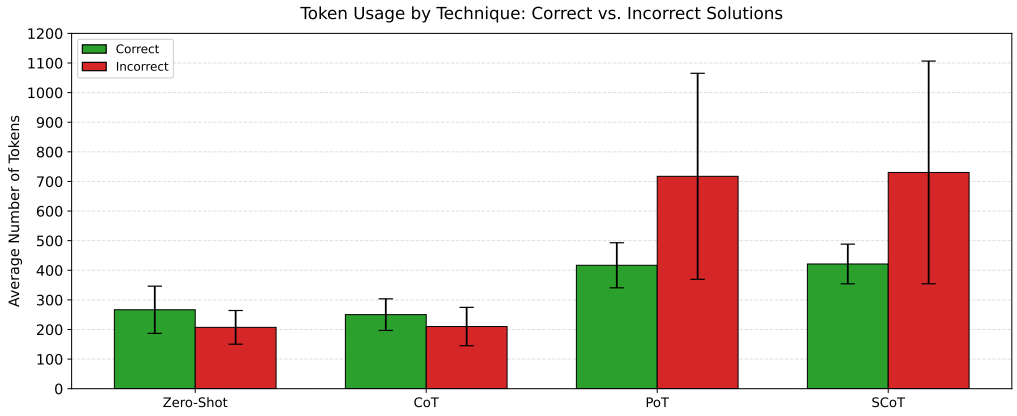


Figure 3. Average number of tokens for correct and incorrect answers.

In PoT, correct answers had an average of approximately 420 tokens, while incorrect answers exceeded 700 tokens. A similar pattern was observed in SCoT, in which correct answers remained close to 420 tokens, whereas incorrect answers reached approximately 730 tokens. Thus, even without considering the additional planning phase,

code-based prompting techniques produced longer chains for incorrect answers than for correct ones. This behavior suggests the presence of "costly errors", that is, answers in which the model consumes more computational resources and produces longer chains, but does not reach the correct answer. Therefore, chain length should be interpreted together with accuracy and reasoning structure, rather than as an isolated indicator of quality.

The presence of longer errors in PoT and SCoT also suggests that more structured techniques may amplify both correct reasoning and incorrect trajectories. When the initial decomposition is adequate, formalization may help consolidate the solution. When the initial decomposition is flawed, the additional structure may only prolong the error, increasing cost without producing benefit.

4.7. Transition Patterns between Reasoning Stages

Figure 4 presents the transition graphs between reasoning stages for each prompt technique. In each graph, nodes represent the stages *DEFINE*, *BLOOM*, *CYCLE*, and *FINAL*, while black dots indicate the beginning and end of the reasoning chain. Directed edges represent observed transitions between stages, and edge labels indicate the corresponding transition probabilities. This representation makes it possible to observe the "grammar" of thought chains, that is, how the model sequentially organizes its reasoning under each prompt strategy.

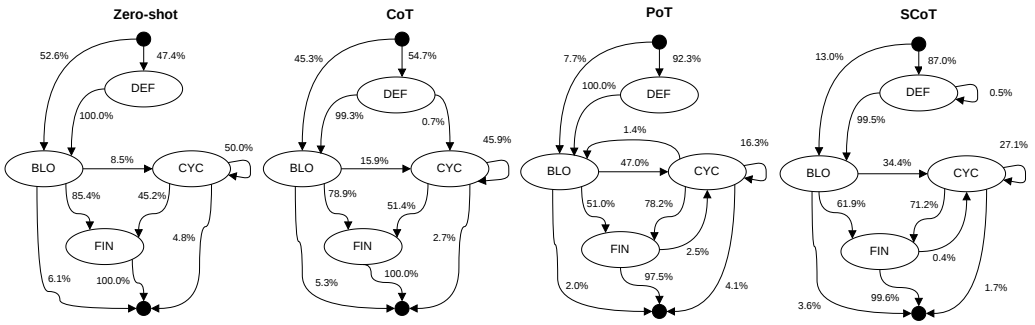


Figure 4. Transition graphs among reasoning stages by prompt technique.

In Zero-shot and CoT, the transition structure is dominated by relatively direct trajectories. Both techniques frequently move from *DEFINE* to *BLOOM* and then from *BLOOM* to *FINAL*, followed by the end of the chain. In Zero-shot, the transition from *BLOOM* to *FINAL* reaches 85.4%, while in CoT it reaches 78.9%. Although both techniques also contain transitions involving *CYCLE*, these transitions are less central to the overall flow than in the code-based prompting strategies.

In PoT and SCoT, the graphs show a stronger participation of *CYCLE*. In PoT, the transition from *BLOOM* to *CYCLE* reaches 47.0%, while in SCoT it reaches 34.4%. Moreover, both techniques present substantial transitions from *CYCLE* to *FINAL*, with 78.2% in PoT and 71.2% in SCoT. This indicates that code-based prompts more frequently induce an intermediate reconstruction or verification stage before the final answer. The presence of self-transitions in *CYCLE*, especially in SCoT, further suggests that some reasoning chains include repeated verification or refinement steps.

The transition analysis complements the previous results. Even when PoT and SCoT achieve statistically similar accuracy, they induce reasoning chains with different

structural characteristics from Zero-shot and CoT. In particular, PoT and SCoT increase the participation of reconstruction cycles, suggesting that PE in LRMs acts not only on final performance, but also on the organization of the reasoning path. Thus, the prompt does not merely alter the probability of correctness; it also modifies the sequence of stages followed by the model toward the answer.

4.8. General Discussion

The quantitative and structural results converge toward the central hypothesis of this work. PE techniques remain relevant in LRMs, but their role differs from that observed in LLMs. In LLaMA, CoT, PoT, and SCoT mainly act as mechanisms for inducing reasoning, with SCoT providing the strongest gains. In DeepSeek, however, CoT does not significantly outperform Zero-shot, while PoT and SCoT improve performance but do not differ significantly from each other. This suggests that, in LRMs, PE operates less as a trigger for reasoning and more as a mechanism for modulating the structure, cost, and formalization of a reasoning process that already emerges naturally.

The qualitative analyses reinforce this interpretation. BLOOM concentrates most of the inferential effort, while reconstruction cycles appear more frequently under structured prompts but do not reliably distinguish correct from incorrect answers. Moreover, SCoT substantially increases token cost without a statistically significant accuracy gain over PoT. These findings indicate that LRM evaluation should combine accuracy with efficiency and structural metrics, since longer or more formal reasoning may represent either productive verification or unproductive rumination.

5. Conclusions and Future Work

In the evaluated setting, DeepSeek-R1-Distill-LLaMA-8B outperformed LLaMA-3.1-8B-Instruct across all PE techniques on GSM8K, with statistically significant gains under Zero-shot, CoT, PoT, and SCoT. However, the role of PE differed across model paradigms. In LLaMA, structured prompts produced broader performance variation and acted mainly as mechanisms for inducing reasoning, with SCoT achieving the best result. In DeepSeek, CoT did not significantly improve over Zero-shot, while PoT and SCoT achieved similar accuracy. This suggests that, in LRMs, PE techniques no longer act only as triggers for reasoning, but also modulate how reasoning is structured, formalized, and executed.

The qualitative analyses complemented the quantitative results by showing that structured techniques altered both the cost and organization of reasoning chains. SCoT produced the highest average number of tokens, whereas PoT achieved similar performance with lower cost, indicating that greater formalization did not necessarily imply higher accuracy. In addition, reconstruction cycles were more frequent under structured prompts but did not reliably distinguish correct from incorrect answers, suggesting that longer reasoning may represent both productive verification and unproductive rumination.

5.1. Threats to Validity and Future Work

This study has limitations that should be considered when interpreting the results. First, although the evaluated models have the same approximate scale and related architecture, they differ in training procedure, distillation, instruction tuning, and possibly training data exposure. Therefore, the observed differences should not be interpreted as the isolated causal effect of RL, but as an empirical comparison between the selected LLM and

LRM. Second, the evaluation used a sample which enabled paired statistical comparisons under fixed computational cost, but it did not capture variance across multiple stochastic generations or across other reasoning benchmarks. Third, correctness was based on the final numerical answer extracted from model outputs, and ambiguous cases required manual inspection. For the qualitative analysis, most reasoning chains were annotated automatically after a manual calibration stage. Although the annotation procedure included a balanced manual audit and inter-annotator agreement assessment, automatic labels may still contain errors. Future work should extend the evaluation to additional benchmarks, multiple random samples, repeated generations, and larger manually validated annotation sets.

References

Chen, W. et al. (2023). Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*.

Cobbe, K. et al. (2021). Training verifiers to solve math word problems.

Guo, D. et al. (2025). Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Jiang, B. et al. (2024). A peek into token bias: Large language models are not yet genuine reasoners. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4722–4756.

Li, J. et al. (2025). Structured chain-of-thought prompting for code generation. *ACM Transactions on Software Engineering and Methodology*, 34(2):1–23.

Marjanovic, S. V. et al. (2026). Deepseek-r1 thoughtology: Let's think about LLM reasoning. *Transactions on Machine Learning Research*.

Mondorf, P. and Plank, B. (2024). Beyond accuracy: Evaluating the reasoning behavior of large language models – a survey.

Muennighoff, N. et al. (2025). s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20286–20332.

OpenAI (2024). Openai o1. `https://openai.com/o1/`. Accessed on: 2025-01-31.

Sahoo, P. et al. A systematic survey of prompt engineering in large language models: Techniques and applications.

Shi, F. et al. (2023). Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR.

Wei, J. et al. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Xu, F. et al. (2025). Toward large reasoning models: A survey of reinforced reasoning with large language models. *Patterns*, 6(10):101370.

Yang, D. et al. Code to think, think to code: A survey on code-enhanced reasoning and reasoning-driven code intelligence in llms.