Zipping the Thought: When and How Compressed Reasoning Data Works in LLM Post-Training

Anonymous ACL submission

Abstract

Large language models (LLMs) can now solve complex problems through long chain-of-thought (CoT) reasoning, but the trade-off between performance and token cost remains a central challenge. To address this issue, supervised fine-tuning (SFT) often uses compressed reasoning data, where CoT traces are shortened into compact forms. However, the effect of such compressed reasoning data on post-training remains poorly understood. In this paper, we propose a taxonomy of CoT consisting of Explicit CoT, which outputs all operations without aggregation, Composed CoT, which combines multiple operations into a single step, and Implicit CoT, which omits intermediate operations. We construct a synthetic compositional reasoning task that allows controlled variation of difficulty, compression granularity, and data size, and conducted a comprehensive set of experiments across different model families and sizes. Notably, we find that (i) coarser CoT requires more SFT data, (ii) compared with Explicit CoT, Composed CoT and Implicit CoT benefit more from data scaling, while Composed CoT benefits from data repetition and Implicit CoT tends to lead to memorization, (iii) unlike SFT, subsequent reinforcement learning (RL) with verifiable rewards (RLVR) decomposes compressed steps learned during SFT, and (iv) unidirectional CoT ordering shows stronger generalization on longer sequential tasks. Our findings provide implications for CoT design under data resource constraints and offer important insights into the mechanisms of SFT and RL in LLM post-training.

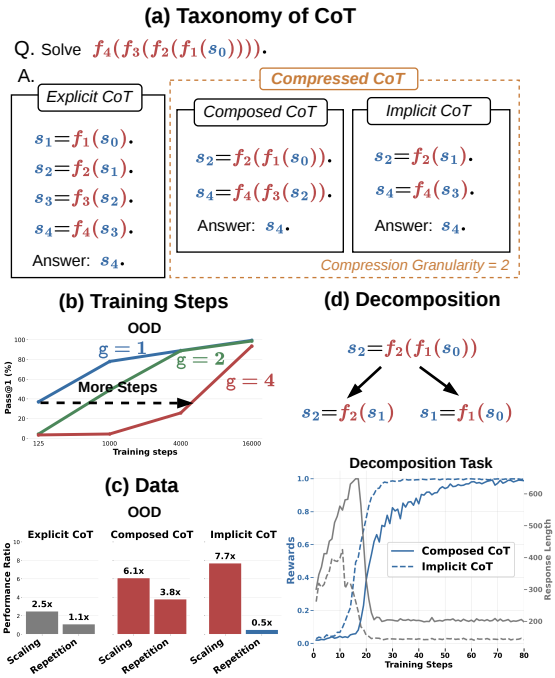


Figure 1: **(a) Taxonomy of CoT.** Example of an op $= 4$ task. f_i denotes an operation and s_i denotes a value. See Figure 2 for task descriptions. **(b) CoT Granularity in SFT and Training Steps.** Coarser-grained CoT SFT requires more training steps. **(c) Data Scaling vs. Data Repetition.** Compressed CoT benefits substantially from data scaling. Composed CoT benefits from data repetition, whereas Implicit CoT is adversely affected by it. **(d) Decomposition of Steps by RLVR.** On-policy exploration in RLVR can decompose compressed reasoning steps. Results in (b), (c), and (d) are from Qwen2.5-3B.

1 Introduction

Algorithmic tasks can be formulated as the sequential composition of simpler subproblems (Bellman, 1957; Newell and Simon, 1972). In large language models (LLMs), such compositional structure is often realized through chain-of-thought (CoT) reasoning (Wei et al., 2022; Kojima et al., 2022), which externalizes intermediate reasoning steps as tokens before producing the final answer. This capability is often enhanced during supervised fine-tuning (SFT) and reinforcement learning (RL) with verifiable rewards (RLVR) at post-training (Jaech et al., 2024; Guo et al., 2025; Lambert et al., 2025).

Despite the remarkable success of CoT reasoning, the widespread deployment of LLM agents has escalated token costs (Bai et al., 2026), moti-

vating efforts to compress reasoning traces without compromising performance.

While prior studies have proposed CoT compression methods, shortening long CoT in into a compact form, such as SFT or self-distillation (Huang et al., 2025; Du et al., 2026) on data with filtered (Li et al., 2026c; Xia et al., 2025) or rewritten tokens (Wu et al., 2025b), it remains unclear how these structurally achieve compression. For instance, CoT compression structures and their impacts on the data volume and training epochs required for SFT remains largely unexplored.

In parallel, the distinct roles of SFT and RL in LLMs (Chu et al., 2025; Matsutani et al., 2026) have received increasing attention. A pessimistic view holds that RL merely sharpens the distribution without discovering novel solutions (Yue et al., 2025), whereas an optimistic view suggests that RL generalizes beyond SFT through unseen composition—combining skills $f_i(x)$ and $f_j(x)$ into $f_i(f_j(x))$ (Anderson, 1982; Arora and Goyal, 2023; Yuan et al., 2026; Park et al., 2025; Cheng et al., 2026). This compositionality is crucial for understanding RL generalization (Chu et al., 2025; Shenfeld et al., 2026). However, to achieve this, it remains unclear whether models can successfully decompose and reconstruct entangled skill chunks observed in imitation data.

An important question to ask is then *(i) how CoT compression impacts the learning dynamics of SFT*, and *(ii) whether SFT and RL can decompose compressed steps*.

To bridge this gap, we define composition in reasoning data as the bundling of multiple atomic operations into a single step, without explicitly decomposing them or emitting intermediate results. We first introduce a taxonomy of CoT. Specifically, we categorize CoT into fully decomposed, Explicit CoT (detailing every operation and value) and Compressed CoT (aggregating them). The latter is further subdivided into Composed CoT, which explicitly lists and executes the combined operations at once, and Implicit CoT, which yields only the final step of the chunk (Figure 1 (a)). Based on this taxonomy, we follow Ye et al. (2025a,b); Zhou et al. (2025); Zhang et al. (2025) to construct an arithmetic synthetic task with controllable difficulty, compression granularity, and CoT types to investigate different CoT data properties.

We employ Qwen2.5 (Yang et al., 2025a) and Llama-3 (Grattafiori et al., 2024) models (from 0.5B to 14B parameters), and analyze the effect of data size, and compression granularity, and CoT types in SFT data on out-of-distribution generalization, measured on a test set of longer compositional tasks (Figure 4), and its downstream impact on RLVR.

We found that SFT on coarsely Compressed CoT requires supervision on a larger number of data points. In particular, Compressed CoT benefits more from data scaling than Explicit CoT. Within Compressed CoT, Composed CoT benefits from data repetition, whereas Implicit CoT suffers from OOD performance degradation. We further observed that SFT cannot decompose compressed steps, while RLVR can do so via exploration, enabling generalization to longer compositional tasks that require decomposition. Finally, by analyzing CoT order, we found that unidirectional CoT generalizes, whereas hierarchically chunked CoT fails to generalize to longer tasks.

We hope these findings will help navigate the trade-off between reasoning length and performance, serving as a guideline for optimal data design under resource constraints. Furthermore, we expect this work to elucidate the critical role of RL in LLM post-training, particularly how skill composition and decomposition drive generalization, ultimately helping post-training discover new solutions through skill composition.

The paper is organized as follows. In Section 2, we conceptually categorize CoT compression and introduce a controlled empirical task. In Section 3.1, we study how compression granularity affects required SFT steps, while in Section 3.2, we analyze data scaling and repetition across different CoT compression. In Section 3.3, we discuss reasoning chain decomposition alongside the comparative roles of SFT and RLVR. In Section 3.4, we show how CoT ordering in SFT data affects generalization. We discuss connections to prior work in Section 4. Detailed related work appears in Appendix A.

2 Problem Setup

In this section, we introduce (a) the taxonomy of CoT for compositional tasks, and (b) the synthetic tasks for experiments.

2.1 Taxonomy of CoT

Compositional, long sequential problems can be expressed as a chain of atomic operations. We can write this chain by nesting a transformation $f_i(\cdot)$

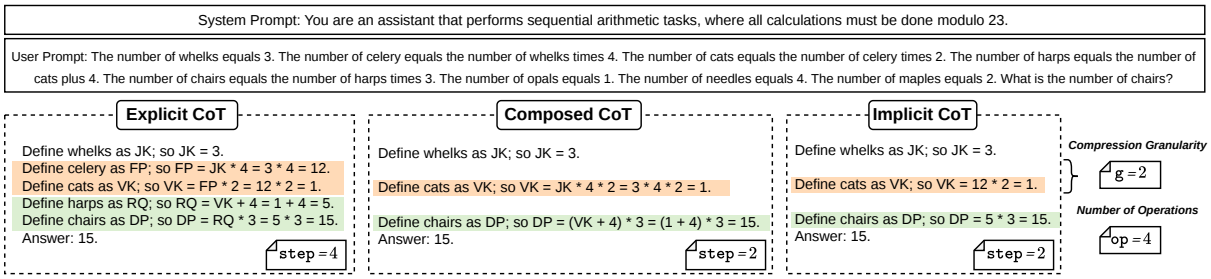


Figure 2: **Synthetic Dataset for Compositional Tasks.** Each question consists of natural language descriptions of inter-parameter relations, including addition, subtraction, and multiplication, with initial parameter values. The task requires sequentially applying the specified operations modulo 23 to infer the value of a target parameter. We use a CoT format of the form: "Define [parameter] as [variable]; so [variable] [operation] = [value]." This example requires 4 operations (op $= 4$) to solve. For compressed CoT (Composed CoT and Implicit CoT), the compression granularity is set to 2 (g $= 2$), so the problem is solved in 2 steps (`step` $= 2$).

i.e., $f_i(f_{i-1}(\ldots(f_1(s_0)))$. For example, in math tasks, this transformation f_i corresponds to mathematical operation that maps from values to calculated results. In knowledge tasks, such as multi-hop factual reasoning (Ho et al., 2020; Trivedi et al., 2022; Press et al., 2023), f_i corresponds to relation between two entities (e.g., "'s capital is" maps from "Japan" to "Tokyo"). In graph tasks (Wang et al., 2023; Sun et al., 2024), f_i corresponds to directed edge that transits from one node (vertex) to another, and in discrete state-action environments that can be described as Markov Decision Process (MDPs), such as games (e.g., spatial navigation tasks (Nolte et al., 2025; Dao and Vu, 2025; Li et al., 2026b) and ARC-AGI (Chollet, 2019)) and robotics tasks, f_i corresponds to an action (e.g., "Up", "Down") that moves the agent (LLM) from one state to another. Hereafter, we refer to f_i as an *operation* and s_i as a *value*.

We then formalize the compression of CoT. Prior works proposed methods to make compressed reasoning data by filtering tokens (Li et al., 2026c; Xia et al., 2025), rewriting reasoning expressions (Wu et al., 2025b), and self-distillation (Huang et al., 2025; Du et al., 2026), but these methods remain heuristic and lack general validity across models and domains. Accordingly, we categorize CoT into *Explicit CoT*, *Composed CoT*, and *Implicit CoT*.

Explicit CoT Explicit CoT processes each operation step by step without aggregating or skipping them: $s_1 = f_1(s_0)$, $s_2 = f_2(s_1)$, ..., where s_i is an (intermediate) value.

Composed CoT In Composed CoT, multiple operations are composed within a single reasoning step. Thus, the model proceeds as $s_2 = f_2(f_1(s_0))$, $s_4 = f_4(f_3(s_2))$, ... Composed CoT omits intermediate values while explicitly specifying all applied operations. In this case, we call the *compression granularity* of CoT is 2 (g $= 2$), two functions are composed into one step, and we control this granularity hereafter. Explicit CoT corresponds to g $= 1$.

Implicit CoT Implicit CoT skips operations, so the model only outputs $s_4 = f_4(s_3)$, and skips $s_1 = f_1(s_0)$, $s_2 = f_2(s_1)$, and $s_3 = f_3(s_2)$, where compressed granularity is 4 (g $= 4$). While compressed CoT outputs all the operations they apply and hide the values, Implicit CoT hides and internally processes both the operations and the values. Implicit CoT can substantially reduce response length, which has motivated methods for internalizing CoT into continuous states (Deng et al., 2023, 2024; Hao et al., 2025; Shen et al., 2025; Wei et al., 2026), but their learning limitations have been noted (Li et al., 2026a).

2.2 Synthetic Dataset for Compositional Tasks

Figure 4: **Train–Test Split by op.** Training is performed on tasks with short op sequences, while tasks with longer op sequences are used for OOD evaluation.

For these compositional reasoning tasks, we construct a testbed that lets us control data size, difficulty, and compression granularity. We employ synthetic arithmetic dataset illustrated in Figure 2. Parameter dependencies are given in the contexts. Each dependency involves a single operation from

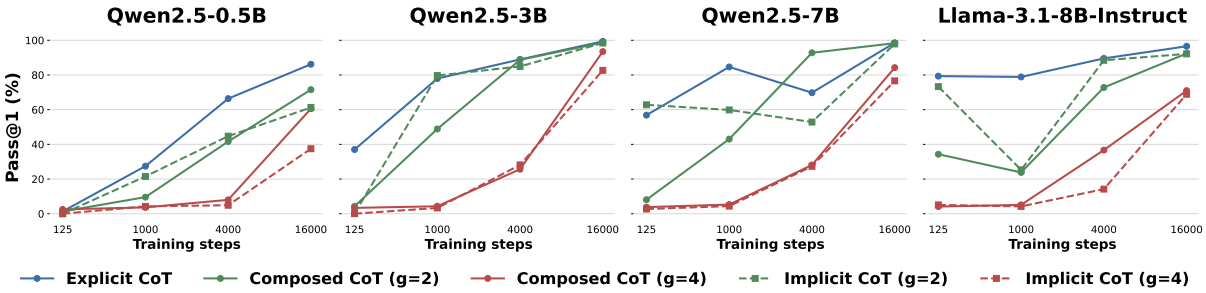


Figure 3: **Compression Granularity and Training Steps.** The bar chart reports the average performance of Qwen2.5-0.5B, 3B, 7B, and Llama-3.1-8B-Instruct at steps 125, 1000, 4000, and 16000 after SFT with Explicit CoT, Composed CoT, and Implicit CoT with $\mathsf{g} = 2, 4$. Models are trained on tasks with $\mathsf{op} = 8, 16, 24$. Evaluation results are averaged over $\mathsf{op} = 32, 40, 48, \ldots, 96, 104$ tasks.

$\{+, -, \times\}$, and we define the number of such operations as the difficulty, denoted by op. We define out-of-distribution (OOD) tasks as those with larger op than in the training set following Zhang et al. (2025) (Figure 4). This arithmetic task is formulated as a "chaining" of many atomic operations with only local dependencies. This dataset is a variant of those in Ye et al. (2025a,b); Zhou et al. (2025); Zhang et al. (2025). Our task corresponds to the special case where the in-degree and out-degree of their computational graph are both 1. In addition, to prevent numerical explosion as op grows, we restrict the operations to modulo 23 following (Ye et al., 2025a,b). To more closely approximate a real-world setting, we follow Zhou et al. (2025) and inject noise parameters into the context. See Appendix B for detailed data generation process.

3 Experiments

The interplays between data points and compressed granularity are addressed in Section 3.1. Data diversity considered in Section 3.2, decomposition of Compressed CoT is investigated in Section 3.3, and difference between CoT orders are addressed in Section 3.4. For details on SFT and RLVR training, please refer to Appendix C.

Notations. op denotes the number of operations required to solve a compositional task. CoT is categorized into *Explicit CoT* and *Compressed CoT*, where the latter includes *Composed CoT* and *Implicit CoT*. For Compressed CoT, g denotes the number of operations grouped into a single step, and `step` denotes the number of steps required to solve the task. Thus, it holds that $\mathsf{op} = \mathsf{g} \times \mathtt{step}$, where $\mathsf{g} = 1$ for *Explicit CoT*.

3.1 Compression Granularity and Training Steps

We first analyze how compressed reasoning data affect SFT, particularly on how the compression granularity influences the number of training steps required for performance improvement.

We perform SFT on Qwen2.5-0.5B, 1.5B, 3B, 7B, 14B, and Llama-3.2-1B, 3B-Instruct and Llama-3.1-8B-Instruct using Explicit CoT, Composed CoT with $\mathsf{g} = 2, 4$, and Implicit CoT on $\mathsf{op} = 8, 16, 24$ tasks. Training is conducted for one epoch, corresponding to 16,000 steps, on 768k samples. Since SFT is performed only on op that are multiples of 8, and multiples of the compression granularities considered here (2 and 4), the model can reach the final answer without fractional operations. We evaluate each model on $\mathsf{op} = 32, 40, 48, \ldots, 104$ tasks, where each task has 336 samples. Throughout the rest of this paper, evaluation is performed with greedy decoding. The maximum number of tokens is set by the op range: 4096, 8192, 12288, and 16384 for op = 25-44, 45-64, 65-84, and 85-104, respectively, ensuring sufficient generation length.

Figure 3 shows performance (Pass@1) averaged over $\mathsf{op} = 32, 40, 48, \ldots, 96, 104$ at each training step for each model and SFT setting. Across all model sizes, larger g in Compressed CoT, including both Composed CoT and Implicit CoT, requires more training steps to reach the same level of performance. One can interpret this as evidence that, to train models to use CoT with higher compression granularity (coarser reasoning traces), one should prepare and perform SFT on more data points. See Appendix D.1 for detailed results.

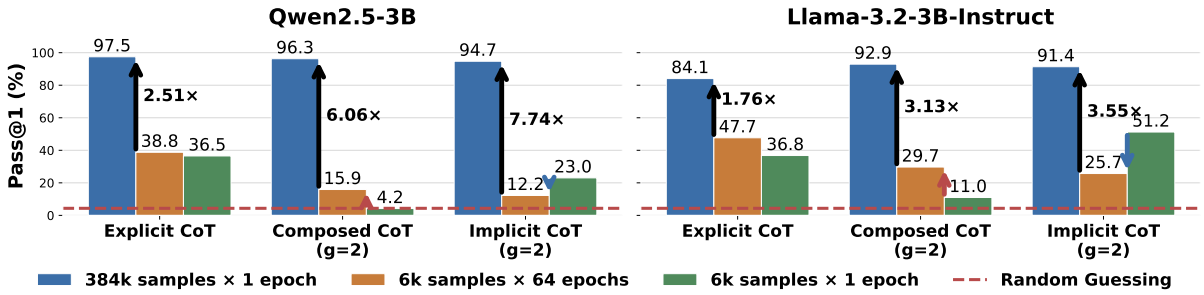


Figure 5: **Data Scaling vs Data Repetition.** The bar chart reports average performance of Qwen2.5-3B and Llama-3.2-3B-Instruct after SFT with Composed CoT and Implicit CoT with $g = 2$. Models are trained with 384k samples for 1 epoch, 6k samples for 64 epochs, and 6k samples for 1 epoch. Evaluation results are averaged over $\mathsf{op} = 32, 40, 48, \ldots, 96, 104$ tasks. Since the computation is performed modulo 23, the chance level ($\frac{1}{23}$), is indicated by the red dashed line.

Takeaway 1. Coarser-grained Compressed CoT requires more data.

3.2 Data Scaling vs Data Repetition

Having analyzed how compression granularity affect the number of SFT training steps required, we next investigate whether data repetition or scaling is more effective for Compressed CoT data. This analysis is motivated by prior work showing that, in long CoT SFT, repeating a small amount of high-quality data can outperform scaling to larger and more diverse datasets (Ye et al., 2025c; Muennighoff et al., 2025; Kopiczko et al., 2026).

We perform SFT on Qwen2.5-3B and Llama-3.2-3B-Instruct under three training settings, with 384k samples for 1 epoch, 6k samples for 64 epochs, and 6k samples for 1 epoch. Note that the settings of 384k samples for one epoch and 6k samples for 64 epochs have the same computational budget. We evaluate each model on $\mathsf{op} = 32, 40, 48, \ldots, 104$ tasks. Comparing 384K samples for 1 epoch with 6K samples for 64 epochs, Composed CoT and Implicit CoT show larger relative accuracy gains from diverse data scaling than Explicit CoT. This suggests that Compressed CoT requires data scaling more than Explicit CoT.

Figure 5 shows performance averaged over $\mathsf{op} = 32, 40, 48, \ldots, 96, 104$ for each model and SFT setting. Since answers are integers in $0, \ldots, 22$ for our modulo 23 arithmetic task, we compare relative performance.

Takeaway 2. Compared with Explicit CoT, Composed CoT benefits more from diverse data than from repeated data under a fixed computational budget.

Additionally, comparing 6K samples for 64 epochs with 6K samples for 1 epoch shows that data repetition improves Composed CoT but degrades Implicit CoT. Composed CoT explicitly represents the operation applied at each step as $f_i(f_{i-1}(\cdot))$. In contrast, Implicit CoT depicts only the latter component, $f_i(\cdot)$. These results suggest that Implicit CoT may be more prone to memorization or overfitting to a specific length (Dziri et al., 2023; Pruthi et al., 2026). One can benefit from SFT with Compressed CoT, which reduces response length, when diverse data are available. In contrast, when data are limited, one should apply data repetition for Composed CoT, whereas such repetition should be avoided for Implicit CoT. See Appendix D.2 for detailed results.

Takeaway 3. SFT on Composed CoT improves performance through data repetition, whereas Implicit CoT exhibits performance degradation.

3.3 Decomposition of Reasoning Chains

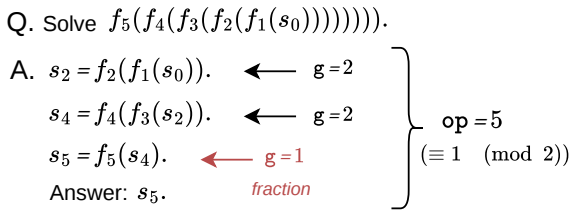
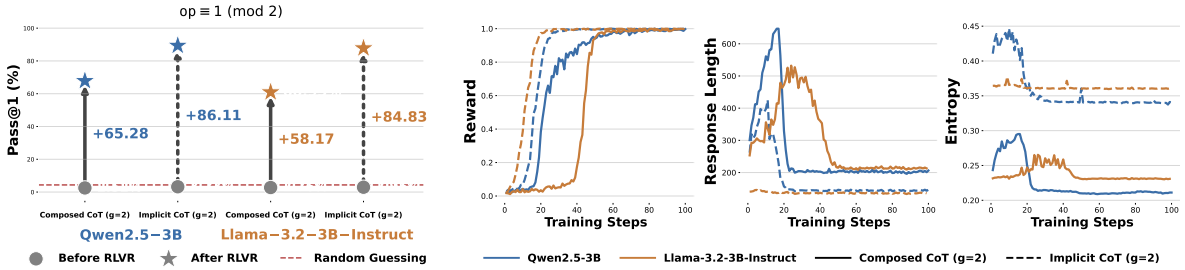


Figure 7: **Illustration of Tasks Requiring Decomposition.** For tasks with $\mathsf{op} = 5$ ($\equiv 1 \pmod 2$), applying CoT with $g = 2$ produces $g = 1$ fractions required to solve the problem.



(a) RLVR Evaluation Results.

(b) RLVR Training Dynamics.

Figure 6: **Decomposition of Composed Steps by RLVR.** (a) Dumbbell plot showing changes in the average evaluation results over $\mathsf{op} = 25, 27, 29, \ldots, 101, 103$ before and after RLVR on $\mathsf{op} = 9, 11, 13, 15$ tasks, using checkpoints obtained by SFT on Qwen2.5-3B and Llama-3.2-3B-Instruct with Composed CoT and Implicit CoT with $\mathsf{g} = 2$. (b) Training dynamics of the mean reward, mean rollout response length, and mean token entropy at each steps. Since the computation is performed modulo 23, the chance level ($\frac{1}{23}$), is indicated by the red dashed line.

We investigate (i) whether SFT can decompose reasoning steps below the compression granularity, and (ii) whether such decomposition can be induced, particularly through subsequent RLVR. Prior work has shown that, for atomic operations (skills) already acquired by LLMs, SFT fails to handle unseen compositions, whereas RL generalizes to them (Yuan et al., 2026; Park et al., 2025; Cheng et al., 2026). However, enabling such composition requires decomposing aggregated operations observed in imitation data and recombining the resulting components (Lake and Baroni, 2018; Kim and Linzen, 2020; Keysers et al., 2020; Hupkes et al., 2021). To this end, we evaluate it on problems that require decomposition, as their computation cannot be represented exactly at the given compression granularity (Figure 7). Specifically, for Qwen2.5-3B and Llama-3.2-3B-Instruct, we perform SFT on CoT traces with $\mathsf{g} = 2$, so that the smallest observed unit is $f_i(f_{i-1}(\cdot))$, while f_i itself is never observed in isolation. We test whether the models can solve problems that require decomposition into individual f_i steps, and whether subsequent on-policy RLVR enables this ability without access to offline decomposed CoT traces.

We train the models for one epoch on 768k samples with $\mathsf{op} = 8, 16, 24$ ($\equiv 0 \pmod 2$), and evaluate them on $\mathsf{op} = 25, 27, 29, \ldots, 101, 103$ ($\equiv 1 \pmod 2$). We then perform GRPO (Shao et al., 2024) on $\mathsf{op} = 9, 11, 13, 15$ ($\equiv 1 \pmod 2$) and evaluate on $\mathsf{op} = 25, 27, 29, \ldots, 101, 103$ tasks.

Figure 8 reports evaluation results after SFT on even and odd op. The results show that, when trained on compressed CoT traces with $\mathsf{g} = 2$, both Composed CoT and Implicit CoT solve OOD tasks with even op but fail on tasks with odd op.

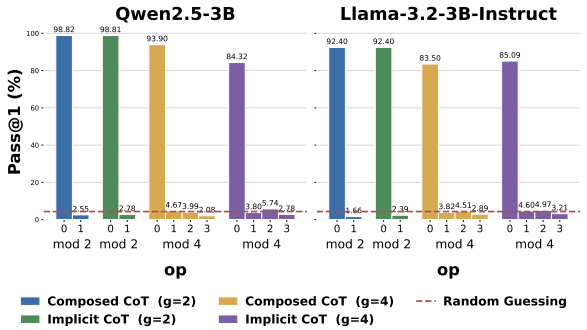


Figure 8: **Decomposition of Composed Steps by SFT.** The bar chart reports the average performance of Qwen2.5-3B and Llama-3.2-3B-Instruct after SFT on 384k samples using Composed CoT and Implicit CoT with $\mathsf{g} = 2, 4$, evaluated on $\mathsf{op} = 25, 26, 27, \ldots, 103, 104$. Results are averaged over op values grouped by residue classes modulo 2 $(0, 1)$, and modulo 4 $(0, 1, 2, 3)$. Since the computation is performed modulo 23, the chance level ($\frac{1}{23}$), is indicated by the red dashed line.

Takeaway 4. SFT cannot decompose reasoning steps below the compression granularity of the training data.

However, applying RLVR to problems with odd op enables the models to solve tasks that were almost unsolved after SFT, where performance remained near chance level. As shown in Figure 6, the models acquire the ability to solve problems that require decomposition. This shows that decomposition can be achieved using only outcome correctness, without preparing CoT traces as in SFT. It also suggests that, with subsequent RLVR, hidden steps learned through compressed SFT data can be used in new compositions. The response

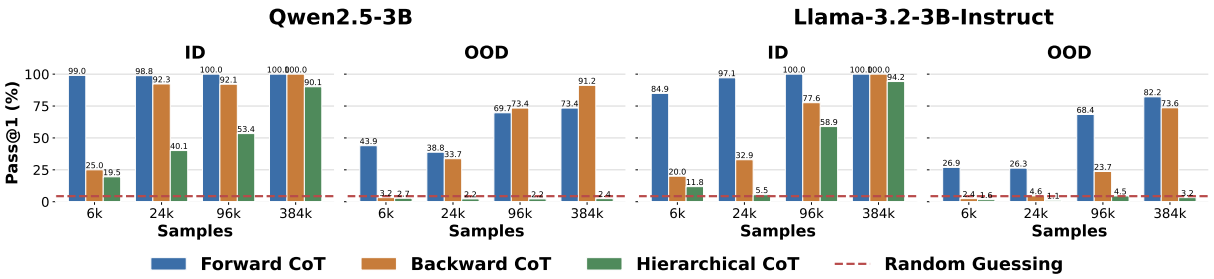


Figure 9: **Effect of CoT Order.** The bar chart reports the average performance of Qwen2.5-3B and Llama-3.2-3B-Instruct after SFT on 384k samples using Forward CoT, Backward CoT, and Hierarchical CoT with op $= 8, 16$, evaluated on op $= 8, 16$ (ID) and op $= 32, 64, 128$ (OOD). Since the computation is performed modulo 23, the chance level ($\frac{1}{23}$), is indicated by the red dashed line.

length dynamics further support this interpretation. Response length increases sharply when the reward first starts to rise, and then decreases as training converges. This phenomenon is not observed when RLVR is applied to even op tasks (Figure 14). This may indicate that the model first decomposes the g $= 2$ compressed reasoning into explicit CoT steps, and then converges to a more efficient strategy that uses g $= 2$ compressed CoT for the main computation and a single step operation (g $= 1$) for the fraction. See Appendix D.3 for examples of reasoning outputs.

Moreover, Implicit CoT requires fewer optimization steps than Composed CoT before the reward starts to increase. Composed CoT exhibits higher average token entropy along trajectories, indicating that it is more strongly constrained by the SFT format of emitting the observed $f_i(f_{i-1}(\cdot))$ chunks under g $= 2$. See Appendix D.3 for detailed results.

Takeaway 5. RLVR enables decomposition thanks to its on-policy exploration. Implicit CoT can be decomposed more efficiently than Composed CoT with RLVR.

3.4 Effect of CoT Order on Generalization

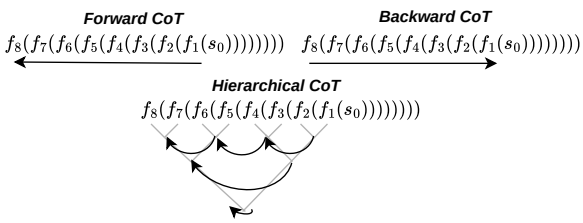


Figure 10: **Illustration of Different CoT Orders.** Forward CoT, Backward CoT, and Hierarchical CoT for tasks with op $= 8$.

We have so far examined compressed CoT mainly for sequential problems such as $f_8(f_7(f_6(f_5(f_4(f_3(f_2(f_1(s_0)))))))))$, where op $= 8$ and CoT follows the forward order $f_1 \to f_2 \to f_3 \to f_4 \to f_5 \to f_6 \to f_7 \to f_8$ (we call this *forward CoT*.) Other reasoning orders are also possible. In *backward CoT*, the model reasons backward from the desired answer, following $f_8 \to f_7 \to f_6 \to f_5 \to f_4 \to f_3 \to f_2 \to f_1$. In *Hierarchical CoT*, the problem is split into chunks (sets of adjunct functions), intermediate results are stored as variables, and these results are composed hierarchically. Intuitively, this corresponds to solving a mathematical problem by evaluating useful subexpressions and then combining them to obtain the final answer. Illustrations of different CoT orders are highlighted in Figure 10.

To quantify how these CoT orders affect SFT, we train for one epoch on 6k, 24k, 96k, and 384k samples with op $= 8, 16$. We evaluate ID performance at op $= 8, 16$ and OOD performance at op $= 32, 64, 128$. For Hierarchical CoT, we set op to powers of two to ensure a binary-tree structure.

Figure 9 shows the average ID and OOD evaluation results for each CoT order. For both models, Hierarchical CoT fails to generalize to OOD (longer compositional tasks), whereas Forward CoT and Backward CoT do generalize. However, Backward CoT requires more data. This suggests that sequential reasoning in a unidirectional order is important for long CoT in Transformer-based LLMs, and that the larger data requirement of Backward CoT during post-training may be related to the data distribution used in pre-/mid-training. Intuitively, non-unidirectional CoT, such as Hierarchical CoT, may require more advanced reasoning and greater working memory, as it must retain intermediate values from multiple chunks and use them in

future operations. When constructing a CoT trace for sequential tasks, one should preferably adopt a unidirectional design in which a single value is used in the subsequent operation. See Appendix D.4 for detailed results.

Takeaway 6. Forward CoT and Backward CoT is more robust on sequential compositional tasks, compared to Hierarchical CoT. Backward CoT requires more data than Forward CoT.

4 Discussion and Related Works

In this work, we study when and how compressed reasoning data affect SFT and subsequent RLVR. We first find that increasing compression granularity requires more training steps and more data to achieve performance gains. Among the CoT formats, Compressed CoT benefits more from data scaling, and within Compressed CoT, Composed CoT benefits from data repetition, whereas Implicit CoT is harmed by it. We further show that, even when models are exposed to compressed reasoning data during SFT, subsequent RLVR exploration can decompose reasoning into atomic operations. Finally, we analyze the ordering of CoT supervision and show that Hierarchical CoT generalizes poorly OOD compared with sequential Forward and Backward CoT, while Backward CoT requires more data than Forward CoT. Below, we discuss connections to related literature, practical implications, and potential directions for future work.

Accuracy and Length Pareto Frontier. As LLM agents are deployed across a wide range of use cases, post-training on compressed reasoning data becomes increasingly important for inference efficiency and token cost (Bai et al., 2026). Our results demonstrate that compressed CoT requires more data points, and that an SFT-then-RL pipeline is effective (Matsutani et al., 2026; Limozin et al., 2026).

Synthetic Reasoning Data. Our results also provide practical guidelines for designing compressed reasoning data under limited resource constraints. For example, when training with short CoT data via SFT but data diversity is limited, it is preferable to train for multiple epochs on Composed CoT, which explicitly includes all operations at a single steps, rather than on Implicit CoT. More broadly, promising directions include designing synthetic data generation pipelines that produce appropriate CoTs from problems (Wu et al., 2025b), as well as rewriting existing reasoning traces (Su et al., 2025; Fujii et al., 2026) into effective compressed formats.

Composition in LLMs. We find that SFT on compressed reasoning data cannot decompose reasoning below the granularity observed in supervision, whereas subsequent RLVR can recover atomic operations and generalize to longer compositional tasks. Unlike SFT, RLVR does not require explicitly decomposed CoT traces. For example, supervision on only $f_2(f_1(\cdot))$ and $f_3(f_2(\cdot))$ may suffice to recover the underlying components f_1, f_2, and f_3, and potentially recombine them to form $f_1(f_3(\cdot))$. In our synthetic task, we consider only three operations, $\{+, -, \times\}$. In real-world domains such as mathematics, code, science, and logical reasoning, however, the space of effective operations is much larger, and often semi-infinite. Because operation combinations grow combinatorially, manually decomposing them during data construction is often impractical, making it important for an RL algorithm to discover useful decompositions from outcome reward supervision alone. This observation echoes prior findings that SFT struggles with unseen compositions, whereas RL can generalize to OOD compositions (Yuan et al., 2026; Park et al., 2025; Cheng et al., 2026). Against recent pessimistic views on RL for LLMs (Yue et al., 2025), our results suggest that RL can break apart coarse reasoning chunks learned from compressed SFT data and recombine them into further compositions, supporting the view that RL can discover new solutions not present in the base model. We believe that composition and decomposition through RL are a promising direction toward LLMs that learn from experience (Silver and Sutton, 2025) with less dependence on data coverage (Chen et al., 2026a; Zhang et al., 2025).

Internal Circuits for Compositional Reasoning. Prior work has studied internal circuits in multi-hop reasoning of LLMs (Li et al., 2024b; Yang et al., 2024; Biran et al., 2024; Yu et al., 2025; Tang et al., 2025; Hong et al., 2026; Yao et al., 2026). It would be interesting to study compositional reasoning in models trained with Explicit CoT versus Compressed CoT, via mechanistic interpretability analyses of internal computation.

Limitations

This study has several limitations. First, our experiments rely on synthetic arithmetic tasks to control the experimental setting, and we do not evaluate on real data. Second, our formulation of CoT reasoning focuses on compositional tasks with nested operations. It does not cover tasks with more diverse structures, such as branching procedures or search. Third, we define OOD evaluation as generalization to longer compositional tasks. This notion of OOD differs from domain shift, where the input distribution changes in content or domain. Finally, our experiments are conducted with Transformer-based Qwen2.5 and Llama 3 models. We do not evaluate whether the findings extend to a broader range of architectures, such as Mamba (Gu and Dao, 2024), GatedDeltaNet (Yang et al., 2025b), hybrid models (Lenz et al., 2025; Hu et al., 2026), and Looped Transformers (Giannou et al., 2023).

Ethical Considerations

This work studies the data and training behavior of LLMs and does not introduce a new deployed system, dataset containing potential risks, or capability intended for harmful use. We therefore believe that the direct and immediate risks of this work are limited.

References

John Anderson. 1982. Acquisition of cognitive skill. *Psychological Review*, 89:369–406.

Sanjeev Arora and Anirudh Goyal. 2023. A theory for emergence of complex skills in language models. *arXiv preprint arXiv:2307.15936*.

Sanjeev Arora and Anirudh Goyal. 2025. Metacognitive reuse: Turning recurring llm reasoning into concise behaviors. *arXiv preprint arXiv:2509.13237*.

Gregor Bachmann and Vaishnavh Nagarajan. 2024. The pitfalls of next-token prediction. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 2296–2318. PMLR.

Longju Bai, Zhemin Huang, Xingyao Wang, Jiao Sun, Rada Mihalcea, Erik Brynjolfsson, Alex Pentland, and Jiaxin Pei. 2026. How do AI agents spend your money? analyzing and predicting token consumption in agentic coding tasks. *arXiv preprint arXiv:2604.22750*.

Richard Bellman. 1957. *Dynamic Programming*. Princeton University Press, Princeton, NJ.

Eden Biran, Daniela Gottesman, Sohee Yang, Mor Geva, and Amir Globerson. 2024. Hopping too late: Exploring the limitations of large language models on multi-hop queries. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14113–14130, Miami, Florida, USA. Association for Computational Linguistics.

Abhranil Chandra, Ayush Agrawal, Arian Hosseini, Sebastian Fischmeister, Rishabh Agarwal, Navin Goyal, and Aaron Courville. 2025. Shape of thought: When distribution matters more than correctness in reasoning tasks. *arXiv preprint arXiv:2512.22255*.

Fan Chen, Audrey Huang, Noah Golowich, Sadhika Malladi, Adam Block, Jordan T. Ash, Akshay Krishnamurthy, and Dylan J Foster. 2026a. The coverage principle: How pre-training enables post-training. In *The Fourteenth International Conference on Learning Representations*.

Howard Chen, Noam Razin, Karthik Narasimhan, and Danqi Chen. 2025. Retaining by doing: The role of on-policy data in mitigating forgetting. *arXiv preprint arXiv:2510.18874*.

Jiaao Chen, Xiaoman Pan, Dian Yu, Kaiqiang Song, Xiaoyang Wang, Dong Yu, and Jianshu Chen. 2024. Skills-in-context: Unlocking compositionality in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 13838–13890, Miami, Florida, USA. Association for Computational Linguistics.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, and Greg Brockman et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Mayee F Chen, Nicholas Roberts, Kush Bhatia, Jue WANG, Ce Zhang, Frederic Sala, and Christopher Re. 2023. Skill-it! a data-driven skills framework for understanding and training language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Weize Chen, Jiarui Yuan, Tailin Jin, Ning Ding, Huimin Chen, Zhiyuan Liu, and Maosong Sun. 2026b. The overthinker's DIET: Cutting token calories with DIfficulty-aware training. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Sitao Cheng, Xunjian Yin, Ruiwen Zhou, Yuxuan Li, Xinyi Wang, Liangming Pan, William Yang Wang, and Victor Zhong. 2026. From atomic to composite: Reinforcement learning enables generalization in complementary reasoning. In *The 1st Workshop on Scaling Post-training for LLMs*.

François Chollet. 2019. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. 2025. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. In *Forty-second International Conference on Machine Learning*.

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

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, Zhiyuan Liu, Hao Peng, Lei Bai, Wanli Ouyang, Yu Cheng, Bowen Zhou, and Ning Ding. 2025. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*.

Xingyu Dang, Christina Baek, Kaiyue Wen, J Zico Kolter, and Aditi Raghunathan. 2025. Weight ensembling improves reasoning in language models. In *Second Conference on Language Modeling*.

Alan Dao and Dinh Bach Vu. 2025. AlphaMaze: Enhancing large language models' spatial intelligence via GRPO. *arXiv preprint arXiv:2502.14669*.

Yuntian Deng, Yejin Choi, and Stuart Shieber. 2024. From explicit cot to implicit cot: Learning to internalize cot step by step. *arXiv preprint arXiv:2405.14838*.

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. 2023. Implicit chain of thought reasoning via knowledge distillation. *arXiv preprint arXiv:2311.01460*.

Yanrui Du, Sendong Zhao, Yibo Gao, Danyang Zhao, Qika Lin, Ming Ma, Jiayun Li, Yi Jiang, Kai He, Qianyi Xu, Bing Qin, and Mengling Feng. 2026. S3-CoT: Self-sampled succinct reasoning enables efficient Chain-of-Thought LLMs. *arXiv preprint arXiv:2602.01982*.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. 2023. Faith and fate: Limits of transformers on compositionality. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Kazuki Fujii, Yukito Tajima, Sakae Mizuki, Masaki Kawamura, Hinari Shimada, Taihei Shiotani, Koshiro Saito, Masanari Oi, Taishi Nakamura, Takumi Okamoto, Shigeki Ishida, Kakeru Hattori, Youmi Ma, Hiroya Takamura, Rio Yokota, Jun Sakuma, and Naoaki Okazaki. 2026. Rewriting pre-training data boosts LLM performance in math and code. In *The Fourteenth International Conference on Learning Representations*.

Kanishk Gandhi, Ayush K Chakravarthy, Anikait Singh, Nathan Lile, and Noah Goodman. 2025. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective STars. In *Second Conference on Language Modeling*.

Kanishk Gandhi, Denise H J Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah Goodman. 2024. Stream of search (sos): Learning to search in language. In *First Conference on Language Modeling*.

Angeliki Giannou, Shashank Rajput, Jy-Yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. 2023. Looped transformers as programmable computers. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 11398–11442. PMLR.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, and Alex Vaughan et al. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, and Xiao Bi et al. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E Weston, and Yuandong Tian. 2025. Training large language models to reason in a continuous latent space. In *Second Conference on Language Modeling*.

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

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Guan Zhe Hong, Nishanth Dikkala, Enming Luo, Cyrus Rashtchian, Xin Wang, and Rina Panigrahy. 2026. A implies b: Circuit analysis in LLMs for propositional logical reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. 2026. ThinkPrune: Pruning long chain-of-thought of LLMs via reinforcement learning. *Transactions on Machine Learning Research*.

Shijing Hu, Jingyang Li, Xingyu Xie, Zhihui Lu, Kim chuan Toh, and Pan Zhou. 2026. GRIFFIN: Effective token alignment for faster speculative decoding. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Maggie Huan, Yuetai Li, Tuney Zheng, Xiaoyu Xu, Seungone Kim, Minxin Du, Radha Poovendran, Graham Neubig, and Xiang Yue. 2025. Does math reasoning improve general llm capabilities? understanding transferability of llm reasoning. *arXiv preprint arXiv:2507.00432*.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*.

Kerui Huang, Shuhan Liu, Xing Hu, Tongtong Xu, Lingfeng Bao, and Xin Xia. 2025. Reasoning efficiently through adaptive chain-of-thought compression: A self-optimizing framework. *arXiv preprint arXiv:2509.14093*.

Yao Huang, Huanran Chen, Shouwei Ruan, Yichi Zhang, Xingxing Wei, and Yinpeng Dong. 2026. Mitigating overthinking in large reasoning models via manifold steering. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. 2021. Compositionality decomposed: how do neural networks generalise? (extended abstract). In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI'20.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, and Alex Carney et al. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations*.

Najoung Kim and Tal Linzen. 2020. COGS: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9087–9105, Online. Association for Computational Linguistics.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc.

Dawid J. Kopiczko, Sagar Vaze, Tijmen Blankevoort, and Yuki M. Asano. 2026. Data repetition beats data scaling in long-CoT supervised fine-tuning. *arXiv preprint arXiv:2602.11149*.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2873–2882. PMLR.

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, and 4 others. 2025. Tulu 3: Pushing frontiers in open language model post-training. In *Second Conference on Language Modeling*.

Barak Lenz, Opher Lieber, Alan Arazi, Amir Bergman, Avshalom Manevich, Barak Peleg, Ben Aviram, Chen Almagor, Clara Fridman, Dan Padnos, Daniel Gissin, Daniel Jannai, Dor Muhlgay, Dor Zimberg, Edden M. Gerber, Elad Dolev, Eran Krakovsky, Erez Safahi, Erez Schwartz, and 42 others. 2025. Jamba: Hybrid transformer-mamba language models. In *The Thirteenth International Conference on Learning Representations*.

Juncai Li, Ru Li, Yuxiang Zhou, Boxiang Ma, and Jeff Z. Pan. 2026a. Chain of thought compression: A theoritical analysis. *arXiv preprint arXiv:2601.21576*.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. 2024a. GSM-plus: A comprehensive benchmark for evaluating the robustness of LLMs as mathematical problem solvers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2961–2984, Bangkok, Thailand. Association for Computational Linguistics.

Weijiang Li, Yilin Zhu, Rajarshi Das, and Parijat Dube. 2026b. Do LLMs build spatial world models? evidence from grid-world maze tasks. In *ICLR 2026 the 2nd Workshop on World Models: Understanding, Modelling and Scaling*.

Zeju Li, Jianyuan Zhong, Ziyang Zheng, Xiangyu Wen, Zhijian Xu, Yingying Cheng, Fan Zhang, and Qiang Xu. 2026c. Making slow thinking faster: Compressing LLM chain-of-thought via step entropy. In *The Fourteenth International Conference on Learning Representations*.

Zhaoyi Li, Gangwei Jiang, Hong Xie, Linqi Song, Defu Lian, and Ying Wei. 2024b. Understanding and patching compositional reasoning in LLMs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 9668–9688, Bangkok, Thailand. Association for Computational Linguistics.

Alexis Limozin, Eduard Durech, Torsten Hoefler, Imanol Schlag, and Valentina Pyatkin. 2026. SFT-then-RL outperforms mixed-policy methods for llm reasoning. *arXiv preprint arXiv:2604.23747*.

Xiaofeng Lin, Hejian Sang, Zhipeng Wang, and Xuezhou Zhang. 2025. Debunk the myth of SFT generalization. *arXiv preprint arXiv:2510.00237*.

Samuel Lippl, Thomas McGee, Kimberly Lopez, Ziwen Pan, Pierce Zhang, Salma Ziadi, Oliver Eberle, and Ida Momennejad. 2025. Algorithmic primitives and compositional geometry of reasoning in language models. *arXiv preprint arXiv:2510.15987*.

Bingbin Liu, Sebastien Bubeck, Ronen Eldan, Janardhan Kulkarni, Yuanzhi Li, Anh Nguyen, Rachel Ward, and Yi Zhang. 2023. TinyGSM: achieving 80% on GSM8k with one billion parameters. In *The 3rd Workshop on Mathematical Reasoning and AI at NeurIPS'23*.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, Yejin Choi, Jan Kautz, and Pavlo Molchanov. 2025. DLER: Doing length penalty right - incentivizing more intelligence per token via reinforcement learning. *arXiv preprint arXiv:2510.15110*.

Wei Liu, Ruochen Zhou, Yiyun Deng, Yuzhen Huang, Junteng Liu, Yuntian Deng, Yizhe Zhang, and Junxian He. 2026. Learn to reason efficiently with adaptive length-based reward shaping. In *The Fourteenth International Conference on Learning Representations*.

Kohsei Matsutani, Shota Takashiro, Gouki Minegishi, Takeshi Kojima, Yusuke Iwasawa, and Yutaka Matsuo. 2026. RL squeezes, SFT expands: A comparative study of reasoning LLMs. In *The Fourteenth International Conference on Learning Representations*.

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

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

Vaishnavh Nagarajan, Chen Henry Wu, Charles Ding, and Aditi Raghunathan. 2025. Roll the dice & look before you leap: Going beyond the creative limits of next-token prediction. In *Forty-second International Conference on Machine Learning*.

Allen Newell and Herbert Alexander Simon. 1972. *Human Problem Solving*. Prentice-Hall.

Niklas Nolte, Ouail Kitouni, Adina Williams, Mike Rabbat, and Mark Ibrahim. 2025. Transformers can navigate mazes with multi-step prediction. *arXiv preprint arXiv:2412.05117*.

Simon Park, Simran Kaur, and Sanjeev Arora. 2025. How does rl post-training induce skill composition? a case study on countdown. *arXiv preprint arXiv:2512.01775*.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore. Association for Computational Linguistics.

Purva Pruthi, Andrew Yuan, Alexander Nicholas D'Amour, and David Jensen. 2026. Why transformers succeed and fail at compositional generalization: Composition equivalence and module coverage.

Qihan Ren, Peng Wang, Ruikun Cai, Shuai Shao, Dadi Guo, Yuejin Xie, Yafu Li, Quanshi Zhang, Xia Hu, Jing Shao, and Dongrui Liu. 2026. Rethinking generalization in reasoning sft: A conditional analysis on optimization, data, and model capability. *arXiv preprint arXiv:2604.06628*.

Hejian Sang, Yuanda Xu, Zhengze Zhou, Ran He, Zhipeng Wang, and Jiachen Sun. 2026. CRISP: Compressed reasoning via iterative self-policy distillation. *arXiv preprint arXiv:2603.05433*.

Abulhair Saparov, Srushti Ajay Pawar, Shreyas Pimpalgaonkar, Nitish Joshi, Richard Yuanzhe Pang, Vishakh Padmakumar, Mehran Kazemi, Najoung Kim, and He He. 2025. Transformers struggle to learn to search. In *The Thirteenth International Conference on Learning Representations*.

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

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. 2025. CODI: Compressing chain-of-thought into continuous space via self-distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 677–693, Suzhou, China. Association for Computational Linguistics.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. 2026. RL's razor: Why online reinforcement learning forgets less. In *The Fourteenth International Conference on Learning Representations*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*.

Parshin Shojaee, Seyed Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. 2026. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

David Silver and Richard S. Sutton. 2025. Welcome to the era of experience. In George Konidaris, editor, *Designing an Intelligence*. MIT Press. Podcast.

Yuda Song, Hanlin Zhang, Carson Eisenach, Sham M. Kakade, Dean Foster, and Udaya Ghai. 2025. Mind the gap: Examining the self-improvement capabilities of large language models. In *The Thirteenth International Conference on Learning Representations*.

Zayne Rea Sprague, Jack Lu, Manya Wadhwa, Sedrick Keh, Mengye Ren, and Greg Durrett. 2026. Skillfactory: Self-distillation for learning cognitive behaviors. In *The Fourteenth International Conference on Learning Representations*.

Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. 2025. Nemotron-CC: Transforming Common Crawl into a refined long-horizon pretraining dataset. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2459–2475, Vienna, Austria. Association for Computational Linguistics.

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

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth International Conference on Learning Representations*.

Cheng Tang, Brenden Lake, and Mehrdad Jazayeri. 2025. An explainable transformer circuit for compositional generalization. *arXiv preprint arXiv:2502.15801*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Gladys Tyen, Hassan Mansoor, Victor Carbune, Peter Chen, and Tony Mak. 2024. LLMs cannot find reasoning errors, but can correct them given the error location. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13894–13908, Bangkok, Thailand. Association for Computational Linguistics.

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2023. Can language models solve graph problems in natural language? In *Thirty-seventh Conference on Neural Information Processing Systems*.

Ru Wang, Wei Huang, Selena Song, Haoyu Zhang, Qian Niu, Yusuke Iwasawa, Yutaka Matsuo, and Jiaxian Guo. 2026. Beyond in-distribution success: Scaling curves of cot granularity for language model generalization. In *The Third Conference on Parsimony and Learning (Proceedings Track)*.

Zengzhi Wang, Fan Zhou, Xuefeng Li, and Pengfei Liu. 2025. OctoThinker: Mid-training incentivizes reinforcement learning scaling. *arXiv preprint arXiv:2506.20512*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Xilin Wei, Xiaoran Liu, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Jiaqi Wang, Xipeng Qiu, and Dahua Lin. 2026. SIM-cot: Supervised implicit chain-of-thought. In *The Fourteenth International Conference on Learning Representations*.

Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. 2026. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base LLMs. In *The Fourteenth International Conference on Learning Representations*.

Fang Wu, Weihao Xuan, Ximing Lu, Zaid Harchaoui, and Yejin Choi. 2025a. The invisible leash: Why rlvr may not escape its origin. *arXiv preprint arXiv:2507.14843*.

Yifan Wu, Jingze Shi, Bingheng Wu, Jiayi Zhang, Xiaotian Lin, Nan Tang, and Yuyu Luo. 2025b. Concise reasoning, big gains: Pruning long reasoning trace with difficulty-aware prompting. *arXiv preprint arXiv:2505.19716*.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. 2025. TokenSkip: Controllable chain-of-thought compression in LLMs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 3351–3363, Suzhou, China. Association for Computational Linguistics.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2025a. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Sohee Yang, Elena Gribovskaya, Nora Kassner, Mor Geva, and Sebastian Riedel. 2024. Do large language

models latently perform multi-hop reasoning? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10210–10229, Bangkok, Thailand. Association for Computational Linguistics.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. 2025b. Gated delta networks: Improving mamba2 with delta rule. In *The Thirteenth International Conference on Learning Representations*.

Xinhao Yao, Ruifeng Ren, Yun Liao, Lizhong Ding, and Yong Liu. 2026. Compositional generalization from learned skills via cot training: A theoretical and structural analysis for reasoning. In *The Fourteenth International Conference on Learning Representations*.

Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. 2025a. Physics of language models: Part 2.1, grade-school math and the hidden reasoning process. In *The Thirteenth International Conference on Learning Representations*.

Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. 2025b. Physics of language models: Part 2.2, how to learn from mistakes on grade-school math problems. In *The Thirteenth International Conference on Learning Representations*.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. 2025c. LIMO: Less is more for reasoning. In *Second Conference on Language Modeling*.

Edward Yeo, Yuxuan Tong, Xinyao Niu, Graham Neubig, and Xiang Yue. 2025. Demystifying long chain-of-thought reasoning in LLMs. In *ICLR 2025 Workshop on Navigating and Addressing Data Problems for Foundation Models*.

Dingli Yu, Simran Kaur, Arushi Gupta, Jonah Brown-Cohen, Anirudh Goyal, and Sanjeev Arora. 2024. SKILL-MIX: a flexible and expandable family of evaluations for AI models. In *The Twelfth International Conference on Learning Representations*.

Zeping Yu, Yonatan Belinkov, and Sophia Ananiadou. 2025. Back attention: Understanding and enhancing multi-hop reasoning in large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 11257–11272, Suzhou, China. Association for Computational Linguistics.

Lifan Yuan, Weize Chen, Yuchen Zhang, Ganqu Cui, Hanbin Wang, Ziming You, Ning Ding, Zhiyuan Liu, Maosong Sun, and Hao Peng. 2026. From f(x) and g(x) to f(g(x)): LLMs learn new skills in RL by composing old ones. In *The Fourteenth International Conference on Learning Representations*.

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

Charlie Zhang, Graham Neubig, and Xiang Yue. 2025. On the interplay of pre-training, mid-training, and rl on reasoning language models. *arXiv preprint arXiv:2512.07783*.

Dylan Zhang, Yufeng Xu, Haojin Wang, Qingzhi Chen, and Hao Peng. 2026a. Good SFT optimizes for SFT, better SFT prepares for reinforcement learning. *arXiv preprint arXiv:2602.01058*.

Shenao Zhang, Donghan Yu, Yihao Feng, Bowen Jin, Zhaoran Wang, John Peebles, and Zirui Wang. 2026b. Learning to reason as action abstractions with scalable mid-Training RL. In *The Fourteenth International Conference on Learning Representations*.

Haoyu Zhao, Simran Kaur, Dingli Yu, Anirudh Goyal, and Sanjeev Arora. 2024. Can models learn skill composition from examples? In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

Yang Zhou, Hongyi Liu, Zhuoming Chen, Yuandong Tian, and Beidi Chen. 2025. GSM-∞: How do your LLMs behave over infinitely increasing reasoning complexity and context length? In *Forty-second International Conference on Machine Learning*.

AI Assistant Usage

We used LLMs to obtain feedback on ideas and experiments, and for rephrasing written text and checking grammar. We also used LLMs for coding and debugging. We did not use LLMs to make claims made in the paper.

A Extended Related Work

RL vs SFT. Yue et al. (2025) evaluates the Pass@k metric (Chen et al., 2021; Song et al., 2025; Dang et al., 2025; Wen et al., 2026; Wu et al., 2025a) and shows that RLVR primarily sharpens the base model distribution rather than expanding beyond its support (Dang et al., 2025; Cui et al., 2025; Matsutani et al., 2026). Wang et al. (2025); Zhang et al. (2026b); Chen et al. (2026a); Zhang et al. (2025) also observed that RLVR is effective when sufficient coverage is already established during mid-training. By contrast, Chu et al. (2025)

has found that RL tends to generalizes to OOD and mitigates catastrophic forgetting, which has been interpreted as finding solutions that minimize distributional divergence (Shenfeld et al., 2026; Chen et al., 2025). For reasoning SFT, data diversity and distributional coverage are argued to drive generalization (Huan et al., 2025; Lin et al., 2025; Ren et al., 2026), while repeated training on high-quality data can also be effective (Ye et al., 2025c; Muennighoff et al., 2025; Kopiczko et al., 2026). Reasoning trajectories matter beyond correctness (Gandhi et al., 2024; Chandra et al., 2025), so does their amenability to RL (Zhang et al., 2026a).

Long CoT and Reasoning Compression. LLMs can enhance their reasoning ability by producing Long CoT (Yeo et al., 2025). Prior work has analyzed several aspects of Long CoT reasoning, including planning (Bachmann and Nagarajan, 2024; Nagarajan et al., 2025), search (Gandhi et al., 2024, 2025; Saparov et al., 2025), self-correction (Huang et al., 2024; Tyen et al., 2024), and granularity (Wang et al., 2026). However, as task complexity increases, Long CoT can suffer from snowballing errors (Dziri et al., 2023; Shojaee et al., 2026) and overthinking (Sui et al., 2025). To address these issues, recent studies have explored methods for compressing reasoning traces, such as filtering tokens (Li et al., 2026c; Xia et al., 2025), prompting teacher models (Wu et al., 2025b), to generate compressed CoT traces for SFT, and for self-distillation (Huang et al., 2025; Du et al., 2026) or on-policy distillation (Sang et al., 2026). Methods have been proposed to internalize CoT traces into continuous states, which are referred to as Latent CoT (Deng et al., 2023, 2024; Hao et al., 2025; Shen et al., 2025; Wei et al., 2026). RL approaches have also been incorporated response length penalties into the reward (Hou et al., 2026; Huang et al., 2026; Chen et al., 2026b; Liu et al., 2026, 2025).

Composition in LLMs. Arora and Goyal (2023) formalized skill compositons in LLMs. Building on this, previous works studied data selection (Chen et al., 2023), in-context prompting (Chen et al., 2024), benchmark (Yu et al., 2024), SFT (Zhao et al., 2024), especially by conditioning behaviors (Arora and Goyal, 2025) and targeting skills (He et al., 2025), and self-distillation (Sprague et al., 2026). Yao et al. (2026) analyzed compositional generalization under distribution shift and Lippl et al. (2025) identified compositional geometry of algorithmic primitives. Recently, Yuan et al. (2026) and Cheng et al. (2026) investigated the effect of RL on compositional abilities, using synthetic tasks with nested functions and synthetic biographical data, respectively, and Park et al. (2025) employed countdown tasks (Gandhi et al., 2024) to explain structure-dependent hierarchy of learnability.

B Synthetic Dataset

Data Generation Framework. To control data size, difficulty, and compression granularity in compositional reasoning tasks, we construct a GSM8K-like (Cobbe et al., 2021) arithmetic task following Liu et al. (2023); Li et al. (2024a); Zhou et al. (2025); Mirzadeh et al. (2025); Zhang et al. (2025). In data generation, we specify the number of operations required to solve the task, denoted by $\mathtt{op}$, and choose the CoT type of the reasoning trace from Explicit CoT, Composed CoT, and Implicit CoT. For Composed CoT and Implicit CoT, which we collectively refer to as Compressed CoT, we additionally specify the compression granularity $\mathtt{g}$, namely the number of operations processed in a single reasoning step. We denote the resulting number of CoT steps by $\mathtt{step}$.

We summarize below the primary sources of randomness in the data generation process.

- Initial value: The value of the first parameter is randomly sampled from $\{1, 2, 3, 4\}$.
- Operation: The dependency between two adjacent parameters is randomly sampled from $\{+, -, \times\}$.
- Parameter name: Parameters are randomly allocated from 200 populations without duplication within the same dataset.
- Variable name: Variables are assigned random two-letter labels (e.g., DL, ML, and AI). There are $26^2 = 676$ possible combinations.
- Number of distractor parameters: The number of distractor parameters is randomly sampled from $\{1, 2, \ldots, \mathtt{op} - 1\}$.
- Distractor parameter value: The value of each distractor parameter is randomly sampled from $\{1, 2, 3, 4\}$.
- Sentence permutation: The problem statement jointly permutes all $\mathtt{op} + 1$ parameters and distractor parameters.

In our experiments, data are mixed so that the number of examples is balanced across the different values of op used for training and evaluation. We also preprocess the data to ensure that there is no duplication between the training and evaluation sets. Moreover, owing to the randomness of the generation process, we can generate a combinatorial space of examples that is far larger than the scale of the SFT data used in this study (up to 768k examples in a single run).

To prevent numerical values from growing explosively when using long operation sequences with a large number of operations op over $\{+, -, \times\}$, we perform all computations modulo 23, following (Ye et al., 2025a). The system prompt is shown below.

System Prompt

```
You are an assistant that performs
    sequential arithmetic tasks, where
    all calculations must be done modulo
    23.
```

For the CoT data format, following (Ye et al., 2025a,b; Zhou et al., 2025), we use the template "Define [parameter] as [variable]; so [variable] [operation] = [value]." Here, f_i in Figure 1 corresponds to the operation, and s_i corresponds to the value. Below, we show concrete examples of the CoT templates used in Section 3.

Example of Problem

```
The number of hats equals 4. The number
    of chairs equals the number of harps
    minus 1. The number of harps equals
    the number of cats times 2. The
    number of blocks equals 3. The number
     of nautiluses equals 1. The number
    of scarves equals the number of
    spruces minus 4. The number of celery
     equals the number of whelks times 3.
     The number of boots equals the
    number of chairs times 2. The number
    of spruces equals the number of
    onions minus 1. The number of whelks
    equals 1. The number of batons equals
     2. The number of maples equals 1.
    The number of onions equals the
    number of boots times 3. The number
    of opals equals 3. The number of
    needles equals 4. The number of cats
    equals the number of celery times 4.
    What is the number of scarves?
```

Explicit CoT.

Example of Explicit CoT

```
Define whelks as JK; so JK = 1. Define
    celery as FP; so FP = JK * 3 = 1 * 3
    = 3. Define cats as VK; so VK = FP *
    4 = 3 * 4 = 12. Define harps as RQ;
    so RQ = VK * 2 = 12 * 2 = 1. Define
    chairs as DP; so DP = RQ - 1 = 1 - 1
    = 0. Define boots as DE; so DE = DP *
     2 = 0 * 2 = 0. Define onions as MP;
    so MP = DE * 3 = 0 * 3 = 0. Define
    spruces as UA; so UA = MP - 1 = 0 - 1
     = 22. Define scarves as TH; so TH =
    UA - 4 = 22 - 4 = 18. Answer: 18.
```

Composed CoT.

Example of Composed CoT (g = 2)

```
Define whelks as JK; so JK = 1. Define
    cats as VK; so VK = JK * 3 * 4 = 1 *
    3 * 4 = 12. Define chairs as DP; so
    DP = VK * 2 - 1 = 12 * 2 - 1 = 0.
    Define onions as MP; so MP = DP * 2 *
     3 = 0 * 2 * 3 = 0. Define scarves as
     TH; so TH = MP - 1 - 4 = 0 - 1 - 4 =
     18. Answer: 18.
```

Example of Composed CoT (g = 4)

```
Define whelks as JK; so JK = 1. Define
    chairs as DP; so DP = JK * 3 * 4 * 2
    - 1 = 1 * 3 * 4 * 2 - 1 = 0. Define
    scarves as TH; so TH = DP * 2 * 3 - 1
     - 4 = 0 * 2 * 3 - 1 - 4 = 18. Answer
    : 18.
```

Example of Composed CoT (g=8)

```
Define whelks as JK; so JK = 1. Define
    scarves as TH; so TH = (JK * 3 * 4 *
    2 - 1) * 2 * 3 - 1 - 4 = (1 * 3 * 4 *
     2 - 1) * 2 * 3 - 1 - 4 = 18. Answer:
     18.
```

Implicit CoT.

Example of Implicit CoT (g = 2)

```
Define whelks as JK; so JK = 1. Define
    cats as VK; so VK = 3 * 4 = 12.
    Define chairs as DP; so DP = 1 - 1 =
    0. Define onions as MP; so MP = 0 * 3
     = 0. Define scarves as TH; so TH =
    22 - 4 = 18. Answer: 18.
```

Example of Implicit CoT (g = 4)

```
Define whelks as JK; so JK = 1. Define
    chairs as DP; so DP = 1 - 1 = 0.
    Define scarves as TH; so TH = 22 - 4
    = 18. Answer: 18.
```

Example of Implicit CoT (g=8)

```
Define whelks as JK; so JK = 1. Define
    scarves as TH; so TH = 22 - 4 = 18.
    Answer: 18.
```

CoT Order. Backward CoT and Hierarchical CoT realize CoT by sequentially updating variables. Although replacing modular arithmetic over modulo 23 with polynomial expansion complicates symbolic algebraic manipulation, this difficulty is specific to the present sequential setting. Future work may extend the analysis to settings with branching structures.

Example of Forward CoT

```
Define whelks as JK; so JK = 1. Define
    celery as FP; so FP = JK * 3 = 1 * 3
    = 3. Define cats as VK; so VK = FP *
    4 = 3 * 4 = 12. Define harps as RQ;
    so RQ = VK * 2 = 12 * 2 = 1. Define
    chairs as DP; so DP = RQ - 1 = 1 - 1
    = 0. Define boots as DE; so DE = DP *
     2 = 0 * 2 = 0. Define onions as MP;
    so MP = DE * 3 = 0 * 3 = 0. Define
    spruces as UA; so UA = MP - 1 = 0 - 1
     = 22. Define scarves as TH; so TH =
    UA - 4 = 22 - 4 = 18. Answer: 18.
```

Example of Backward CoT

```
Define whelks as PE; so PE = 1. Define
    scarves as SU; so SU = LY - 4. Define
     spruces as LY; so SU = LY - 4 = (PY
    - 1) - 4 = PY - 5. Define onions as
    PY; so SU = PY - 5 = (TD * 3) - 5 = 3
    TD - 5. Define boots as TD; so SU = 3
    TD - 5 = 3(UD * 2) - 5 = 6UD - 5.
    Define chairs as UD; so SU = 6UD - 5
    = 6(KF - 1) - 5 = 6KF - 11. Define
    harps as KF; so SU = 6KF - 11 = 6(BP
    * 2) - 11 = 12BP - 11. Define cats as
     BP; so SU = 12BP - 11 = 12(QO * 4) -
     11 = 2QO - 11. Define celery as QO;
    so SU = 2QO - 11 = 2(PE * 3) - 11 = 6
    PE - 11 = 6(1) - 11 = 18. Answer: 18.
```

Example of Hierarchical CoT

```
Define whelks as PE; so PE = 1. Define
    celery as QO; so QO = 1 * 3 = 3.
    Define cats as BP; so BP = QO * 4 = 3
     * 4 = 12. Define harps as KF; so KF
    = BP * 2 = 2BP. Define chairs as UD;
    so UD = KF - 1 = 2BP - 1. Define
    boots as TD; so TD = UD * 2 = 2UD.
    Define onions as PY; so PY = TD * 3 =
     2UD * 3 = 6UD. Define spruces as LY;
     so LY = PY - 1. Define scarves as SU
    ; so SU = LY - 4 = (PY - 1) - 4 = PY
    - 5. UD = 2BP - 1 = 2 * 12 - 1 = 0.
    SU = PY - 5 = 6UD - 5. SU = 6UD - 5 =
     6 * 0 - 5 = 18. Answer: 18.
```

C Experimental setup

Model We use Qwen2.5 (Yang et al., 2025a) and Llama 3 (Grattafiori et al., 2024), both of which are decoder-only dense Transformers. We use these models in accordance with their respective licenses and terms of use: Qwen2.5 is released under the Apache 2.0 license, and Llama 3 is released under the Llama 3 Community License.

Training Unless noted otherwise, all experiments are conducted on NVIDIA H100 and GH200 GPUs. We use a batch size of 48 for each optimization step. SFT is performed using LLaMA-Factory (Zheng et al., 2024), with the configuration shown in Table 1. For RLVR, we use GRPO (Shao et al., 2024) implemented in the verl framework (Sheng et al., 2024), with hyperparameters listed in Table 2.

Table 1: SFT Configuration.

Component	Setting
Effective batch size	48
Optimizer	AdamW
Learning rate	2.0×10^{-5}
Weight decay	0.1
Max gradient norm	1.0
Scheduler	Cosine
Warmup ratio	0.05
Minimum learning rate	3.0×10^{-6}
Mixed precision	bfloat16

Evaluation. We evaluate each model on op $= 32, 40, 48, \ldots, 104$ tasks, where each task has 336 samples. Evaluation is performed with greedy decoding (temperature $= 0$ and top_p $= 1$). The

Table 2: RLVR Configuration.

Component	Setting
Rollouts	8
Sampling temperature	1.0
Top-p	1.0
Maximum response length	2048
Training batch size	256
Actor learning rate	1.0×10^{-6}
Weight decay	0.01
Maximum gradient norm	1.0
KL coefficient	0.001

maximum number of new tokens is set according to the op range: 4096 for op $= 25$–44, 8192 for op $= 45$–64, 12288 for op $= 65$–84, and 16384 for op $= 85$–104, which provide enough generation budget for correct solutions.

Prompt Template. We use the following prompt templates for Qwen2.5 models and Llama-3 models, respectively.

Prompt Template for Qwen2.5 Models

```
<|im_start|>system
{system}<|im_end|>
<|im_start|>user
{user}<|im_end|>
<|im_start|>assistant
{assistant}<|im_end|>
```

Prompt Template for Llama3 Models

```
<|begin_of_text|><|start_header_id|>
    system<|end_header_id|>

{system}<|eot_id|><|start_header_id|>user
    <|end_header_id|>

{user}<|eot_id|><|start_header_id|>
    assistant<|end_header_id|>

{assistant}<|eot_id|>
```

D Detailed Results

In this section, we report the detailed results for Section 3.

D.1 SFT Results on Different CoT Datasets

For Qwen2.5-0.5B, 1.5B, 3B, 7B, and 14B, we use Explicit CoT, Composed CoT with g $= 2, 4, 8$, and Implicit CoT with g $= 2, 4, 8$. For Llama-3.2-1B-Instruct, Llama-3.2-3B-Instruct, and Llama-3.1-8B-Instruct, we use Explicit CoT, Composed CoT with g $= 2, 4$, and Implicit CoT with g $= 2, 4$. Under each setting, we perform SFT on op $= 8, 16, 24$ ($\equiv 0 \pmod 8$) tasks, and evaluate on op $= 32, 40, 48, \ldots, 96, 104$ ($\equiv 0 \pmod 8$) tasks. Figure 11 and Figure 12 show the evaluation results at checkpoints obtained after one epoch with 6K, 24K, 96K, and 384K training samples.

D.2 SFT Results on Different Number of Epochs.

For Qwen2.5-3B and Llama-3.2-3B-Instruct, we consider Explicit CoT, Composed CoT with g $= 2, 4$, and Implicit CoT with g $= 2, 4$. We perform SFT on op $= 8, 16, 24$ ($\equiv 0 \pmod 8$) tasks under three training regimes: one epoch on 384K samples, 64 epochs on 6K samples, and one epoch on 6K samples. Figure 13 shows the evaluation results on op $= 32, 40, 48, \ldots, 96, 104$ ($\equiv 0 \pmod 8$ tasks).

D.3 RLVR Results

For Qwen2.5-3B and Llama-3.2-3B-Instruct, we start from checkpoints obtained by one-epoch SFT on 24K samples with op $= 8, 16, 24$, using Explicit CoT, Composed CoT with g $= 2$, and Implicit CoT with g $= 2$. We then perform RLVR with GRPO on op $= 9, 11, 13, 15$ and op $= 10, 12, 14$ tasks using the configuration in Table 2. Figure 15 shows the evaluation results on op $= 32, 40, 48, \ldots, 96, 104$ (op $\equiv 0 \pmod 8$) tasks.

Figure 14 shows the training dynamics of RLVR. In terms of response length, op $= 10, 12, 14$ exhibit nearly constant lengths throughout training. By contrast, for op $= 9, 11, 13, 15$, the response length starts to increase when the reward begins to improve, and subsequently converges. This indicates that the model first decomposes chunks obtained from exploration with g $= 2$ into smaller chunks with g $= 1$. Eventually, it performs CoT with g $= 2$ while processing fractions with g $= 1$.

Below are reasoning samples after RLVR under each setting. When RLVR is applied to checkpoints after SFT on Composed CoT, the model continues reasoning with g $= 2$, Qwen2.5-3B outputs with g $= 1$ at the end, and Llama-3.2-3B-Instruct inserts a dummy $*$ 1 at the end. When RLVR is applied to checkpoints after SFT on Implicit CoT, the model outputs only the operation following g $= 2$, and the final step indicates that it is handling a fraction

operation (g = 1).

Qwen2.5-3B, Composed CoT (g = 2)

```
Define whelks as YH; so YH = 3. Define
    cats as YO; so YO = YH + 1 + 4 = 3 +
    1 + 4 = 8. Define chairs as ZD; so ZD
     = YO * 4 * 2 = 8 * 4 * 2 = 18.
    Define onions as YL; so YL = ZD * 3 -
     2 = 18 * 3 - 2 = 6. Define scarves
    as YP; so YP = YL * 3 * 2 = 6 * 3 * 2
     = 13. Define batons as ZQ; so ZQ =
    YP + 2 + 2 = 13 + 2 + 2 = 17. Define
    hats as ZT; so ZT = ZQ * 3 + 1 = 17 *
     3 + 1 = 6. Define maples as YX; so
    YX = ZT - 4 + 3 = 6 - 4 + 3 = 5.
    Define violets as YO; so YO = YX * 3
    * 2 = 5 * 3 * 2 = 7. Define flutes as
     YL; so YL = YO * 3 * 2 = 7 * 3 * 2 =
     19. Define markers as YH; so YH = YL
     * 4 * 3 = 19 * 4 * 3 = 21. Define
    aspens as YL; so YL = (YH - 1) * 3 =
    (21 - 1) * 3 = 14. Define wolves as
    YL; so YL = YL - 2 - 2 = 14 - 2 - 2 =
     10. Define YL + 2 = 10 + 2 = 12.
    Answer: 12.
```

Qwen2.5-3B, Implicit CoT (g = 2)

```
Define whelks as ZD; so ZD = 3. Define
    cats as ZF; so ZF = 4 + 4 = 8. Define
     chairs as ZU; so ZU = 9 * 2 = 18.
    Define onions as ZL; so ZL = 8 - 2 =
    6. Define scarves as ZC; so ZC = 18 *
     2 = 13. Define batons as ZY; so ZY =
     15 + 2 = 17. Define hats as ZN; so
    ZN = 5 + 1 = 6. Define maples as ZD;
    so ZD = 2 + 3 = 5. Define violets as
    ZF; so ZF = 15 * 2 = 7. Define flutes
     as ZD; so ZD = 21 * 2 = 19. Define
    markers as ZD; so ZD = 7 * 3 = 21.
    Define aspens as ZD; so ZD = 20 * 3 =
     14. Define wolves as ZD; so ZD = 12
    - 2 = 10. Define tokens as ZD; so ZD
    = 10 + 2 = 12. Answer: 12.
```

Llama3.2-3B-Instruct, Composed CoT (g = 2)

```
Define whelks as ZD; so ZD = 3. Define
    cats as ZT; so ZT = ZD + 1 + 4 = 3 +
    1 + 4 = 8. Define chairs as ZQ; so ZQ
     = ZT * 4 * 2 = 8 * 4 * 2 = 18.
    Define onions as ZD; so ZD = ZQ * 3 -
     2 = 18 * 3 - 2 = 6. Define scarves
    as ZO; so ZO = ZD * 3 * 2 = 6 * 3 * 2
     = 13. Define batons as ZD; so ZD =
    ZO + 2 + 2 = 13 + 2 + 2 = 17. Define
    hats as ZT; so ZT = ZD * 3 + 1 = 17 *
     3 + 1 = 6. Define maples as ZD; so
    ZD = ZT - 4 + 3 = 6 - 4 + 3 = 5.
    Define violets as ZT; so ZT = ZD * 3
    * 2 = 5 * 3 * 2 = 7. Define flutes as
     ZD; so ZD = ZT * 3 * 2 = 7 * 3 * 2 =
     19. Define markers as ZT; so ZT = ZD
     * 4 * 3 = 19 * 4 * 3 = 21. Define
    aspens as ZD; so ZD = (ZT - 1) * 3 =
    (21 - 1) * 3 = 14. Define wolves as
    ZT; so ZT = ZD - 2 - 2 = 14 - 2 - 2 =
     10. Define tokens as ZD; so ZD = ZT
    * 1 + 2 = 10 * 1 + 2 = 12. Answer:
    12.
```

Llama3.2-3B-Instruct, Implicit CoT (g = 2)

```
Define whelks as ZD; so ZD = 3. Define
    cats as ZC; so ZC = 4 + 4 = 8. Define
     chairs as ZK; so ZK = 9 * 2 = 18.
    Define onions as ZU; so ZU = 8 - 2 =
    6. Define scarves as ZC; so ZC = 18 *
     2 = 13. Define batons as ZD; so ZD =
     15 + 2 = 17. Define hats as ZU; so
    ZU = 5 + 1 = 6. Define maples as ZC;
    so ZC = 2 + 3 = 5. Define violets as
    ZD; so ZD = 15 * 2 = 7. Define flutes
     as ZD; so ZD = 21 * 2 = 19. Define
    markers as ZD; so ZD = 7 * 3 = 21.
    Define aspens as ZD; so ZD = 20 * 3 =
     14. Define wolves as ZD; so ZD = 12
    - 2 = 10. Define tokens as ZD; so ZD
    = 10 + 2 = 12. Answer: 12.
```

D.4 SFT Results on Different CoT Orders

For Qwen2.5-3B and Llama-3.2-3B-Instruct, we consider Forward CoT, Backward CoT, and Hierarchical CoT. We perform SFT on op $= 8, 16$ tasks ($\equiv 0 \pmod 8$), varying the training dataset size among 6k, 24k, 96k, and 384k. Figure 16 shows the evaluation results on op $= 32, 64, 128$ ($\equiv 0 \pmod 8$) tasks.

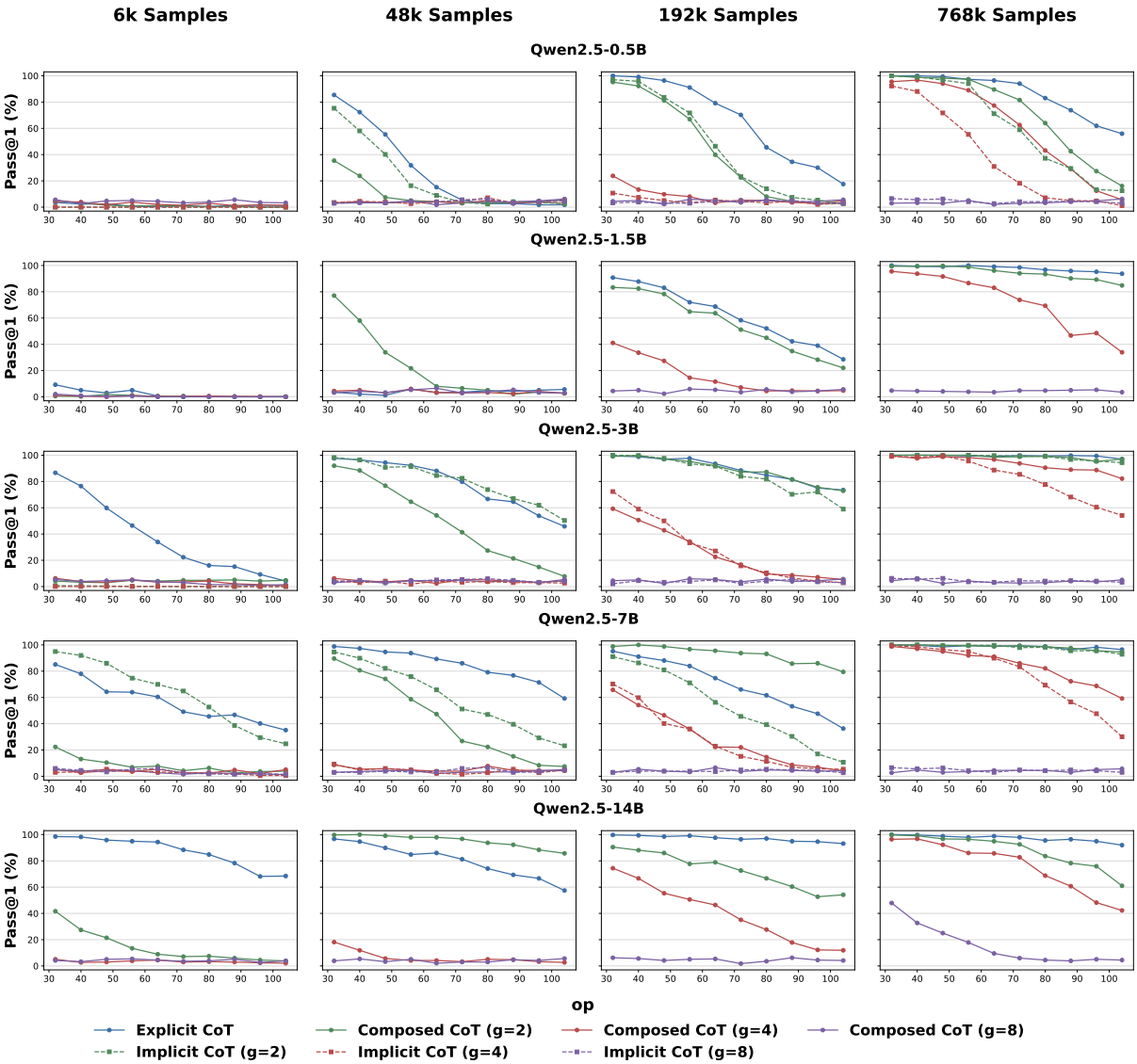


Figure 11: **Evaluation Results of Qwen2.5 Models.** Evaluation results on op $= 32, 40, 48, \ldots, 96, 104$ tasks for checkpoints after SFT for one epoch with 6k, 48k, 192k, and 768k samples for each CoT type at op $= 8, 16, 24$ tasks.

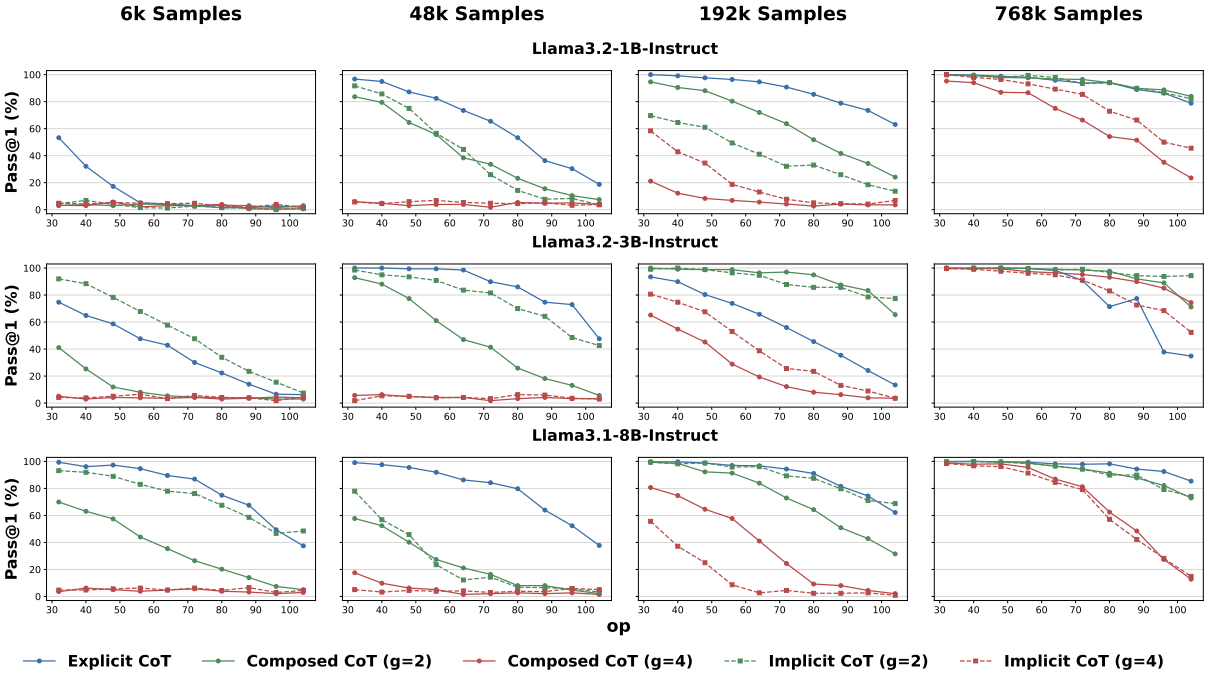


Figure 12: **Evaluation Results of Llama-3 Models.** Evaluation results on $\texttt{op} = 32, 40, 48, \ldots, 96, 104$ tasks for checkpoints after SFT for one epoch with 6k, 48k, 192k, and 768k samples for each CoT type at $\texttt{op} = 8, 16, 24$ tasks.

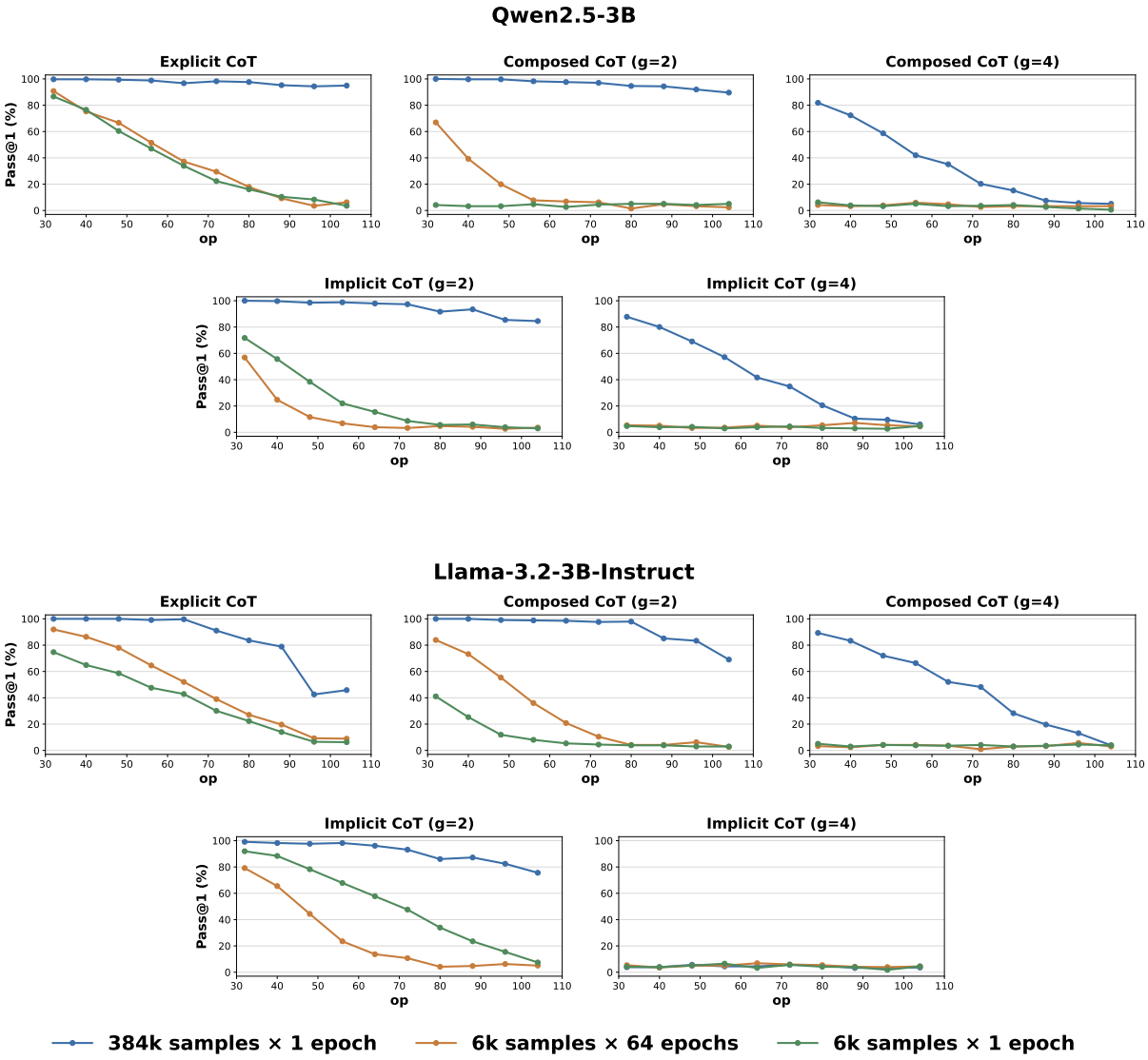
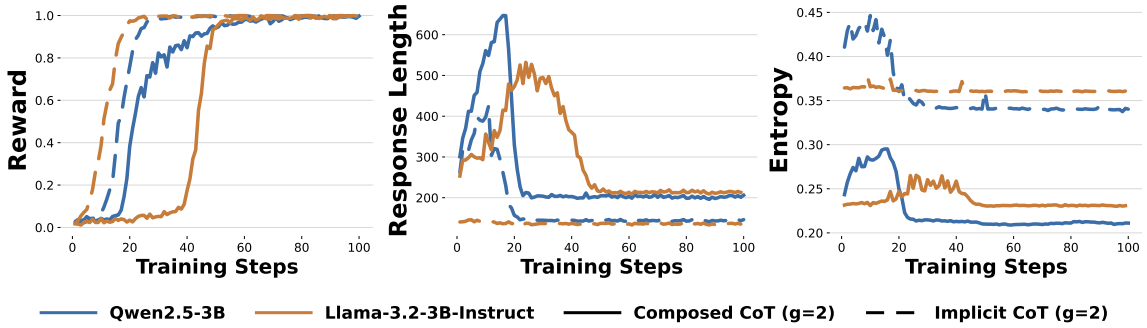
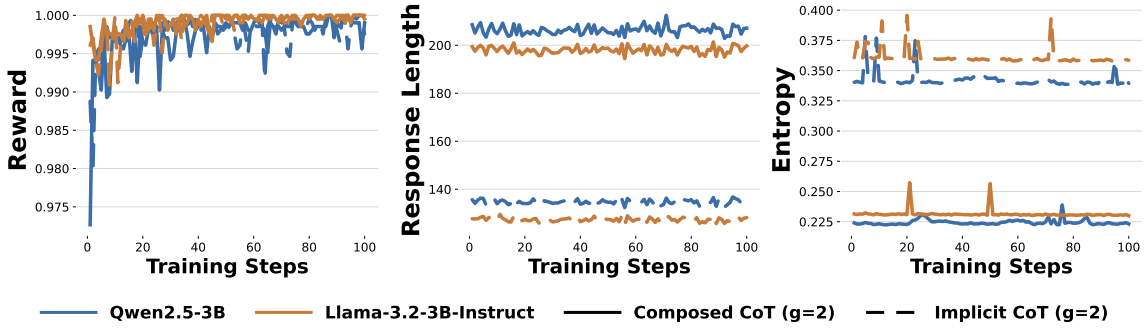


Figure 13: **Evaluation Results of Different Number of Epochs.** Evaluation results on $\mathsf{op} = 32, 40, 48, \ldots, 96, 104$ for checkpoints after SFT for one epoch with 384k samples for one epoch, 6k samples for 64 epochs, and 6k samples for one epoch for each CoT type at $\mathsf{op} = 8, 16, 24$ tasks.



(a) Odd op tasks



(b) Even op tasks

Figure 14: **Evaluation results before and after RLVR on odd and even op tasks.** Training dynamics of the mean reward, mean rollout response length, and mean token entropy at each steps. Odd op task are op $= 9, 11, 13, 15$ and even op task are op $= 10, 12, 14$.

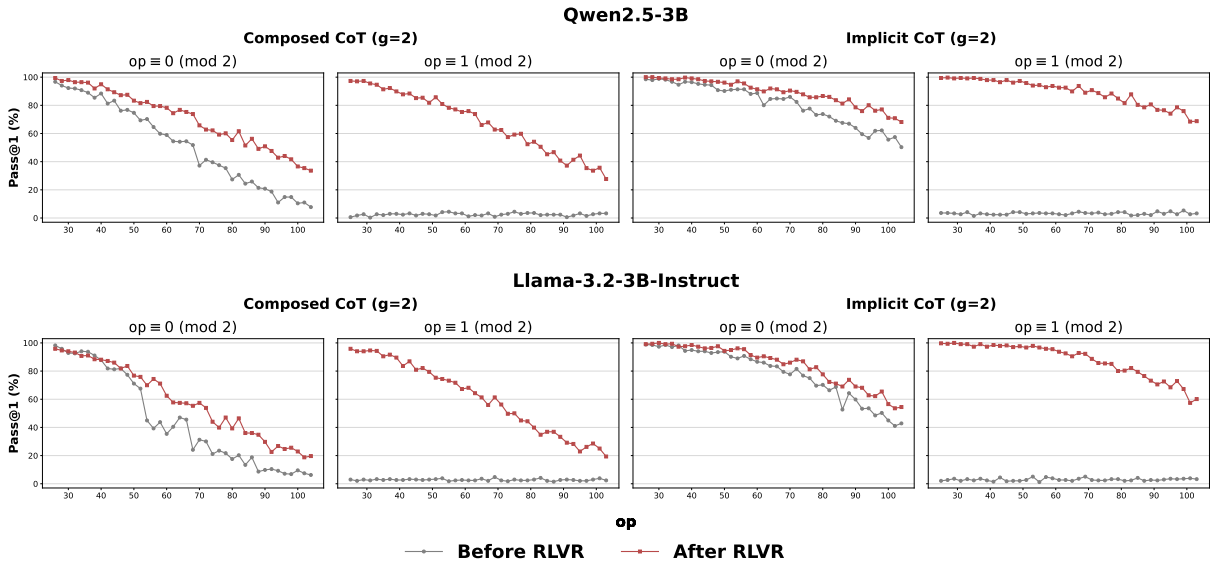


Figure 15: **Evaluation Results of Before and After RLVR on odd op tasks.** Evaluation results on op $= 25, 27, 29, \ldots, 101, 103$ tasks for checkpoints after RLVR on op $= 9, 11, 13, 15$ tasks for each CoT type.

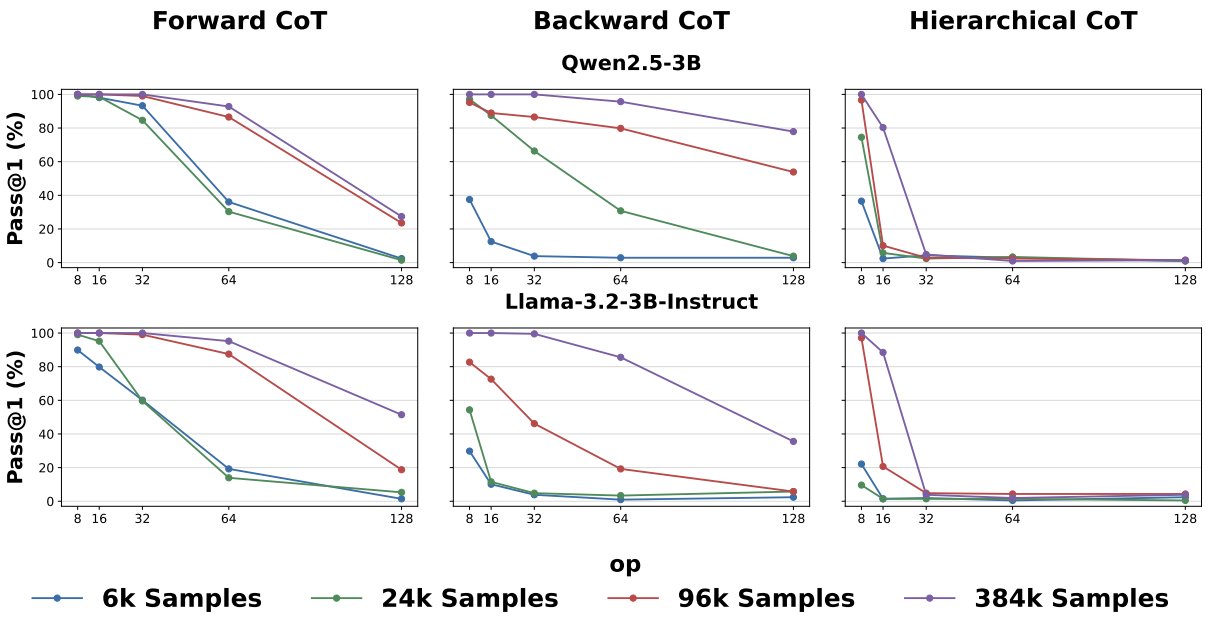


Figure 16: **Evaluation Results of Different CoT Orders.** Evaluation results on $\mathsf{op} = 8, 16, 32, 64, 128$ tasks for checkpoints after SFT on $\mathsf{op} = 8, 16$ tasks for each CoT order. $\mathsf{op} = 8, 16$ tasks are ID, and $\mathsf{op} = 32, 64, 128$ tasks are OOD.