

Learning Composable Chains-of-Thought

Fangcong Yin[♣], Zeyu Leo Liu^{♠♡}, Liu Leqi[♠], Xi Ye[◇], Greg Durrett[♣]

[♣]New York University, [♠]The University of Texas at Austin

[♡]Salesforce AI Research, [◇]Princeton University

fy666@nyu.edu

Abstract

A common approach for teaching large language models (LLMs) to reason is to train on chains-of-thought (CoTs) of in-distribution reasoning problems. However, such annotated data is costly to obtain for every problem of interest. We want reasoning models to generalize beyond their training distribution, and ideally to generalize compositionally: they should combine atomic reasoning skills to solve harder unseen tasks. In this paper, we introduce a method to enable generalization to a target compositional task that has no labeled CoT data. We minimally modify CoT formats of constituent atomic tasks to be **composable**: we add prefixes to CoTs during training time, thereby encouraging the trained model to generate sequences of CoTs during inference. We train individual models on the atomic tasks with composable CoT data and combine them with multitask learning or model merging to address the target compositional task zero-shot. This model can be further trained on a small amount of compositional data using rejection sampling fine-tuning (RFT). Training LLMs on Composable CoT outperforms multitask learning and continued fine-tuning baselines on three domains of controlled compositional tasks and also yields improvement in math skill composition setting derived from the MATH dataset.¹

1 Introduction

Large language models (LLMs) are successful by virtue of the massive amounts of data they are trained on, which makes a wide range of complex problems in-distribution. However, these models still fail at challenging reasoning tasks, particularly when these reasoning tasks are out-of-distribution. Ideally, we want models that can *generalize* to new settings, especially by applying basic “skills”

learned during training in novel combinations to solve problems at inference time. How to empower LLMs with this capability, also called compositional generalization (Piantadosi and Aslin, 2016; Werchan et al., 2015; Conklin et al., 2021; Dziri et al., 2023), remains an open question. For instance, large reasoning models (QwenTeam, 2025; Guha et al., 2025), built on pre-trained LLMs, are typically trained on a large amount of data annotated with chain-of-thought (CoT) traces, but still fall short at generalizing to harder problems than those they were trained on (Sun et al., 2024; Hase et al., 2024; Abreu et al., 2025; Shojaee et al., 2025; Sun et al., 2025) even when they leverage CoT reasoning.

We explore the setting of compositional reasoning where pre-trained LLMs are fine-tuned on CoT data of simple reasoning tasks (atomic tasks) and then evaluated on the *unseen* combinations of them (compositional tasks) with no or limited compositional supervision. We find that models trained with atomic CoT data demonstrate limited generalization to compositional settings. As illustrated in Figure 1, we propose a simple modification of the CoT format of the atomic task training data, which we call **Composable CoT**: we add “proxy prefixes,” which are random filler strings, to the prompt. Then we train models to reason about atomic tasks in the context of the proxy prefixes. Crucially, this training encourages models to chain sequences of CoTs together, which directly enables compositional generalization on a certain class of problems: if a model can chain together a CoT for skill A with a CoT for skill B, it can solve a compositional task requiring skills A and B.

We first experiment with *zero-shot* combination of Composable CoT models with two different approaches: first, merging models trained on individual atomic CoT tasks, and second, multitask learning across our atomic CoT datasets. Such combined models achieve zero-shot compositional

¹Code and data are available at: https://github.com/fc2869/composable_cot.

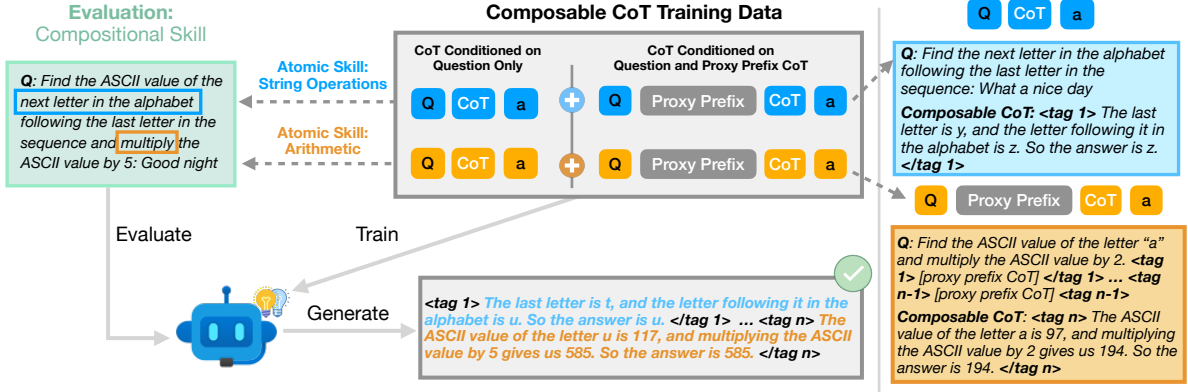


Figure 1: A compositional task involving two atomic skills. We use a data augmentation scheme called **Composable CoT**: we add “proxy prefix CoTs,” such as random filler strings, to the prompt to simulate compositional distributions where a CoT is conditionally generated from the question and other CoTs. At inference time, our model can generate a CoT combining two atomic skills, despite only training on each in isolation.

generalization without seeing compositional data during training.

We then demonstrate that our zero-shot models can be improved further by rejection sampling fine-tuning (RFT) on a limited amount of compositional supervision. Using *only final answer* supervision, our models achieve better compositional behavior.

On controlled compositional tasks involving core reasoning capabilities of LLMs, including natural language skill composition, string manipulation, and arithmetic, our approach outperforms multi-task learning and continued fine-tuning baselines within a given budget of training data. Combining atomic models trained with Composable CoT is better than combining Standard CoT models across different compositional tasks. Composable CoT works to combine both pairs and triples of skills in our experiments. Moreover, Composable CoT can improve math reasoning using skill composition: training models on Composable CoT data of skills for particular mathematical subjects leads to an average improvement of 4.3% over the base model on the MATH (Hendrycks et al., 2021) dataset.

The main contributions of this work include: (1) A novel data augmentation scheme for training CoT models on basic reasoning skills to enable future composition of them for more complicated reasoning tasks. (2) A method for improving compositional reasoning with LLMs by first training models on atomic CoTs with such augmentation and then performing rejection sampling finetuning.

2 Preliminaries

LLM reasoning with chain-of-thought. Given a prompt q that states a reasoning problem with ground truth answer a , an LLM M reasons with chain-of-thought by generating a response that includes a chain-of-thought trace t followed by a predicted answer $\tilde{a}$. Recent works show that supervised fine-tuning LLMs on CoT traces leads to strong reasoning models (Muennighoff et al., 2025; Guha et al., 2025). We define a dataset for a task $\mathcal{T}$ with CoT traces of size N as a set of (prompt, CoT, answer) triples: $D_{\mathcal{T}}^{\text{CoT}} = \{(q, t, a)\}$.

Atomic and compositional tasks. Consider a set of tasks that represent basic reasoning skills, which we call **atomic** tasks. We define **compositional** tasks $\mathcal{T}_A$ as those tasks that can be expressed as a composition of n atomic tasks: $\mathcal{T}_A = g(A)$ where $A = \{\mathcal{T}_i, \dots, \mathcal{T}_j\}$, $|A| = n$ and g is some function to combine the n atomic tasks. We discuss more details for g in Appendix A.

Compositional reasoning from atomic CoT. We assume access to atomic CoT data $D_A^{\text{CoT}} = \{D_{\mathcal{T}_i}^{\text{CoT}} | \mathcal{T}_i \in A\}$. Models fine-tuned on a subset of D_A^{CoT} are **atomic CoT models**.

For their composition $\mathcal{T}_A$, we only have access to a training dataset $D_{\mathcal{T}_A}$ of size $N_{\mathcal{T}_A}$. We make two data assumptions following considerations about how compositional data would work in practice: (1) **Answer only:** The data only contains the answers as labels and *not* labeled CoT traces. This reflects that high-quality annotated CoT supervision may be harder to obtain in practice than correct answers; (2) **Limited compositional supervision:** We assume $N_{\mathcal{T}_A}$ is small. We may be able to collect a

q Question: Multiply the ASCII value of "a" by 2.
 t_1 <tag 1> pwolfub oiaen aaksebnab zldjxhl pasm jdfodf </tag 1>
 $\dots$ [Additional proxy prefix CoTs] ...
 t_{n-1} <tag n-1> dsadu gaulksd pasjfb jfmmaj hsjdn jdkap </tag n-1>
 t_n <tag n> The ASCII value of the letter a is 97, and [...] Answer:
 a 194 </tag n>
 a Answer: 194

Figure 2: Construction of Composable CoT data. We insert $n - 1$ proxy prefixes, shown here as sequences of randomly sampled letters, at the end of the prompt, before the CoT.

small amount of data for each new compositional task of interest, but there are too many possible compositional tasks to collect large-scale data on.

3 Composable Chains-of-Thought

We assume the CoT traces in each of the n atomic task datasets follow a certain distribution distinct to that dataset. A pre-trained LLM M_0 fine-tuned on the atomic CoT data can be seen as a mixture model: it can generate CoT traces from each of those n distributions, but it is unclear whether such models can produce compositional CoTs for compositional tasks. We observe that without additional supervision signals, such fine-tuned models typically only replicate one of the learned atomic reasoning patterns in the generated CoT; we show empirical evidence in Section 7.

To compose n atomic CoTs in one sequence $t_1 \dots t_{n-1} t_n$, the model must allocate substantial probability to $p(t_1 \dots t_{n-1} t_n \mid q)$, when the model is never trained on a sequence of CoTs. Our goal is to augment the atomic CoT training data (q, t, a) into $(q, \text{proxy prefixes}, t, a)$, such that **the training data looks more in-distribution to the compositional data while not explicitly training models on compositional examples**.

3.1 Composable CoT Training Data

Consider an atomic CoT dataset $D_{\mathcal{T}}^{\text{CoT}} = \{(q, t, a)\}$ for $\mathcal{T} \in A$; we call this **standard CoT** data. We augment it with a set of *chain-of-thought tags* $\mathcal{P} = \{p_k\}$ for $k \in \{1, \dots, n\}$.

Proxy Prefixes. Our goal is to augment standard CoT data such that atomic CoT models can learn a proxy distribution that simulates the distribution of the composition of atomic skills, despite not seeing compositional data. Thus, we append proxy prefixes to the prompt to simulate conditional generation of a CoT given other CoTs. In principle, these sampled strings should closely resemble

CoT traces t that will show up in the test distribution. We do not assume knowledge of specific task compositions needed at inference time, but we can imagine sampling reasoning traces for problems related to the atomic task in question, or sampling reasoning-like strings from pre-training data.

Surprisingly, the approach we found to perform best is to use random-length sequences of randomly sampled letters as proxy prefixes. Appendix B shows comparison to more realistic variants. One advantage of random letters is that they do not introduce bias towards any particular distribution of reasoning modes or related tasks, suggesting that learning the structure of Composable CoT is what drives generalization in our settings.

Data Construction. We sample a value of k for each training example $d = (q, t, a)$, and we treat t as the k -th step in a notional compositional reasoning process. To achieve this, we append $k - 1$ proxy prefixes $(t'_1 \dots t'_{k-1})$ to the end of the prompt: $t'_i = \langle \text{tag } i \rangle t_i \langle \text{tag } i \rangle$ for $1 \leq i \leq k - 1$ and t_i is the i -th proxy prefix. By doing so, we obtain the augmented example $d' = (qt'_1 \dots t'_{k-1}, t_k)$ where $t_k = \langle \text{tag } k \rangle t \langle \text{tag } k \rangle$.

Figure 2 illustrates the procedure when $k = n$. The standard CoT t is: “The ASCII value of the letter a is 97, and [...]” We augment the example by: (1) Appending $n - 1$ proxy prefixes to the end of the question q to obtain the augmented prompt $qt'_1 \dots t'_{n-1}$, with each proxy prefix wrapped in a tag; (2) Wrapping the CoT and the answer in a different tag $\langle \text{tag } n \rangle$ as t_n .

We use the scheme above to augment each example in the standard CoT dataset and obtain the augmented dataset $D_{\mathcal{T}}^{\text{aug}}$. At inference time, we do not know a given atomic CoT will be used in which part of the compositional reasoning trace. Because CoT traces in $D_{\mathcal{T}}^{\text{aug}}$ can simulate any of the k -th positions, models trained on $D_{\mathcal{T}}^{\text{aug}}$ should be compatible with compositions of *arbitrary order* instead of priming to any particular order seen during training.

Learning Objective. Then, we fine-tune M_0 on $D_{\mathcal{T}}^{\text{aug}}$ with a supervised fine-tuning objective: $\mathcal{L}_{D_{\mathcal{T}}^{\text{aug}}}(\theta) = \frac{1}{N} \sum_{d' \in D_{\mathcal{T}}^{\text{aug}}} \mathcal{L}_{d'}(\theta)$ where $\mathcal{L}_{d'}(\theta) = -\log p_{\theta}(t_k \mid q \dots t'_{k-1})$. In other words, for each augmented example, we minimize the negative log likelihood of generating the CoT and answer, conditioned on the question and the $(k - 1)$ proxy prefixes.

Note that when $k = 1$, d' does not have any

proxy prefix in the augmented prompt, so the model learns to generate CoT traces conditioned only on the question on those examples (e.g., the top right example in Figure 1). This simulates the scenario where an atomic CoT serves as the initial step of the compositional reasoning. For $1 < k \leq n$, the model learns to generate CoT conditioned on both the question and proxy prefixes (e.g., the bottom right example in Figure 1).

Instantiation of Tags. In practice, models only need to learn differentiations between the n -th tag, which marks the end of the notional n -way compositional reasoning, and all the other tags, which mark intermediate steps. Thus, we set $p_n = \langle \text{suffix} \rangle$, and all other $(n - 1)$ tags as $\langle \text{prefix} \rangle$. Despite only having two instantiations of the tag, this scheme supports training of any length of n of compositional CoT.

The scheme can also generalize to n -way composition *at inference time*. Specifically, for $n > 2$, we can generate a CoT, then append the $\langle \text{suffix} \rangle$ tag, continue to generate, and repeat $(n - 1)$ times, thereby achieving test-time generalization to n -way composition. Details can be found in Section 6.1.

3.2 Combining Atomic CoT Models

After training an atomic CoT model on a single task $\mathcal{T}$, we combine multiple atomic CoT models to perform compositions. We consider two methods.

ComposableCoT-MTL. We apply multitask learning (MTL) to fine-tune M_0 on the combined dataset of $D_A^{\text{aug}} = \sum_{\mathcal{T}_i \in A} D_{\mathcal{T}_i}^{\text{aug}}$ and obtain a single MTL model M_{comb} that learns all atomic tasks.

ComposableCoT-Merge. Model merging is another way to combine multiple models into a single multi-task model (Matena and Raffel, 2022; Yadav et al., 2023). For each $\mathcal{T}_i \in A$, we start from M_0 and fine-tune a model M_i (parametrized by θ_i) on $D_{\mathcal{T}_i}^{\text{aug}}$. Then we use Task Arithmetic (Ilharco et al., 2023) to merge the n models into a single model M_{comb} parametrized by θ_{comb} as a linear combination of the deltas between each fine-tuned model parameter and the base model parameter: $\theta_{\text{comb}} = \theta_0 + \sum_{\mathcal{T}_i \in A} \alpha_i(\theta_i - \theta_0)$ where α is the scaling factor.

Inference. When running zero-shot inference on the compositional task, we append $\langle \text{tag } 1 \rangle$ to the end of the prompt and sample a response from M_{comb} . Then, we append the next tag to the end of the generated response, continue generation, and repeat by appending tags up to $\langle \text{tag } n \rangle$, as we assume that the number of compositions at test

time is known.

Algorithm 1 Rejection Sampling Fine-tuning on Composable CoT Models

Input: The combined model M_{comb} ; dataset $D_{\mathcal{T}_A} = \{(\mathbf{q}_v, a_v)\}_{v=1}^{N_A}$; the number of iterations c .

Output:

```

1:  $M_0 \leftarrow M_{\text{comb}}$  ▷ Initialization
2: for  $w$  in  $1 \dots c$  do
3:   if use rationalization then ▷ Rationalization
4:      $(\tilde{\mathbf{t}}_v, \tilde{a}_v) \leftarrow M_{w-1}(\mathbf{q}_v, a_v) \forall v \in \{1, \dots, N_A\}$ 
5:   else
6:      $(\tilde{\mathbf{t}}_v, \tilde{a}_v) \leftarrow M_{w-1}(\mathbf{q}_v) \forall v \in \{1, \dots, N_A\}$ 
7:   end if
8:    $D_{\text{RFT}} \leftarrow \{(\mathbf{q}_v, \tilde{\mathbf{t}}_v, a_v) \text{ s.t. } v \in \{1, \dots, N_A\} \text{ and } \tilde{a}_v = a_v\}$  ▷ CoTs with correct answers
9:    $M_w \leftarrow \text{SFT}(M_0, D_{\text{RFT}})$  ▷ Fine-tune the combined model on the accepted CoT data
10: end for

```

3.3 Improving Composition with Rejection Sampling Fine-tuning

M_{comb} can be further improved with self-taught reasoning (Zelikman et al., 2022) by rejection sampling fine-tuning (RFT) (Dong et al., 2023; Yuan et al., 2024). Recall that for the compositional task, we only have answer labels instead of CoT traces. M_{comb} can serve as a starting point for RFT where we fine-tune M_{comb} with its own correct CoT responses using the limited compositional data.

Algorithm 1 shows the algorithm. Concretely, we sample responses from M_{comb} for each example in the compositional training data. Using the direct answer labels to verify the sampled responses, we can collect a supervised fine-tuning dataset D_{RFT} to continued fine-tune M_{comb} . Such a process can be repeated for multiple iterations. For open-ended generation tasks that are hard to verify the correctness of sampled outputs only based on answer labels, we follow Zelikman et al. (2022); Ye and Durrett (2022) to perform rationalization to obtain D_{RFT} ; details can be found in Appendix D.3.

4 Experimental Setup

We select controlled evaluation tasks with the following criteria: (1) **Atomic tasks reflect core LLM reasoning skills:** We select atomic tasks that are representative of core skills that span logical, arithmetic, and writing. Prior work (Wei et al., 2022; Dziri et al., 2023; Yu et al., 2024) has shown that these skills can reflect more complicated capabilities such as advanced math reasoning and creative writing; (2) **Atomic skills are distinguishable:** To ensure controlled experiments of compositional generalization, atomic skills need to be dis-

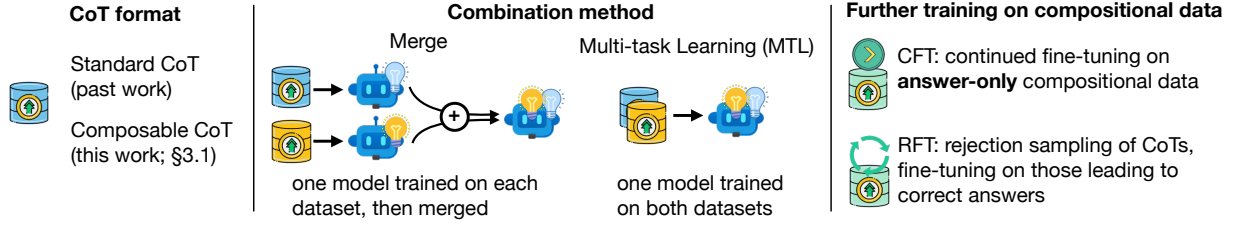


Figure 3: Summary of settings for methods evaluated. Names in the results table reference configurations described in this figure; e.g., ComposableCoT-Merge uses ComposableCoTs with model merging, and in the zero-shot setting does not use further tuning.

Methods	Next Letter + Mult EM	Concat + Next Letter EM	Concat + Mult EM	Skill-Mix Literary + Rhetorical Full Marks	Skill Fraction
Llama 2-7B					
Few-shot Answer	1.0	0.0	0.0	4.1	16.4
Few-shot CoT	2.0	3.0	1.0	7.3	23.1
StandardCoT-Merge	2.0	12.5	2.3	11.0	31.6
ComposableCoT-Merge (Ours)	16.0	19.1	3.0	19.6	37.1
StandardCoT-MTL	5.0	0.0	0.0	17.6	38.7
ComposableCoT-MTL (Ours)	18.7	6.5	3.1	22.9	49.9
Qwen 2.5-7B					
Few-shot Answer	2.4	0.0	2.7	34.7	56.0
Few-shot CoT	2.0	0.0	21.3	31.8	41.6
StandardCoT-Merge	70.4	54.8	77.0	29.8	48.0
ComposableCoT-Merge (Ours)	95.4	19.2	75.4	39.6	62.1
StandardCoT-MTL	3.6	60.9	72.1	42.0	58.2
ComposableCoT-MTL (Ours)	96.3	63.3	74.3	49.0	66.7

Table 1: Zero-shot compositional generalization of ComposableCoT with different combination approaches vs. baselines. *Without any compositional supervision*, using model merging or multitask learning to combine atomic CoT models trained on Composable CoT data outperforms baselines across settings and models.

tinguished from each other so that learning one skill affects learning another skill as little as possible; (3) **Compositions are unseen during pretraining**: General reasoning tasks such as math word problems feature examples that are common in pre-training. Our tasks are less observed, allowing us to attribute the task performance to the efficacy of training approaches rather than better recall of pretraining data.

Our tasks involve string manipulation, arithmetic, and natural language skill composition. Each setting involves atomic tasks and compositional tasks. We ensure that all atomic tasks are learnable through supervised fine-tuning with a small amount of training data ($N_{\mathcal{T}} \leq 500$) as shown in Appendix F. We also confirm that the selected compositional tasks are less frequently seen for pre-trained LLMs: Appendix G shows the high perplexity of the task datasets, and Table 1 shows the low accuracy of few-shot prompting.

String manipulation and arithmetic tasks. We

consider the following atomic tasks. (1) **Next letter in alphabet** (Efrat et al., 2023; Edman et al., 2024); (2) **Letter concatenation** (Wei et al., 2022; Zhou et al., 2023); (3) **ASCII multiplication** (Dziri et al., 2023; Gambardella et al., 2024). We consider the following compositions of two of the atomic tasks: (1) **Next letter + multiplication**; (2) **Concatenation + next letter**; (3) **Concatenation + multiplication**. Details of tasks and data can be found in Appendix C.

Natural language skills. We adapt the compositional benchmark Skill-Mix (Yu et al., 2024): Given the definition and an example of a language skill (e.g. hyperbole), the model writes a sentence to demonstrate the skill about a given topic. We consider an atomic task to be handling skills over a *category* of skills, and we evaluate on two categories: literary devices (*Literary*) and rhetorical devices (*Rhetorical*). Atomic CoT traces for Skill-Mix are distilled from GPT-4o (OpenAI et al., 2024), following Zhao et al. (2024). The composi-

Category	Method	Next Letter + Mult EM	Concat + Next Letter EM	Concat + Mult EM	Skill-Mix Literary + Rhetorical Full Marks	Skill Fraction
Llama 2-7B						
SFT	SFT on Base Model	3.1	5.0	9.0	35.5	60.1
	CFT on StandardCoT-Merge	2.0	16.0	14.0	44.1	65.1
	CFT on StandardCoT-MTL	3.0	26.0	11.0	38.0	62.1
MTL	StandardCoT + Comp Answer	5.0	46.0	13.3	22.9	45.5
RFT	StandardCoT-Merge	0.0	23.0	29.7	26.1	52.0
	ComposableCoT-Merge (Ours)	72.0	46.0	40.0	45.3	66.6
RFT	StandardCoT-MTL	10.1	0.0	0.0	38.8	58.1
	ComposableCoT-MTL (Ours)	23.4	69.6	8.3	44.1	63.5
Qwen 2.5-7B						
SFT	SFT on Base Model	-	31.9	2.0	35.5	60.3
	CFT on StandardCoT-Merge	-	41.1	9.3	51.0	71.4
	CFT on StandardCoT-MTL	-	60.3	12.7	34.7	56.3
MTL	StandardCoT + Comp Answer	-	65.1	7.1	41.2	55.3
RFT	StandardCoT-Merge	-	83.3	83.6	51.8	69.0
	ComposableCoT-Merge (Ours)	-	33.9	87.0	56.7	70.1
RFT	StandardCoT-MTL	-	82.1	89.0	44.9	63.4
	ComposableCoT-MTL (Ours)	-	86.9	88.4	57.6	71.5

Table 2: Compositional task performance of rejection sampling fine-tuning (RFT) upon merged Composable atomic CoT models and other baselines. *Mult* stands for ASCII multiplication and *concat* stands for letter concatenation. *SFT* stands for supervised fine-tuning with the compositional answer data; *CFT* stands for continued fine-tuning; *MTL* stands for multitask learning method. Results on Next Letter + Mult are omitted because the zero-shot performance saturates. For Qwen2.5-7B, RFT on ComposableCoT variants achieves the best compositional performance using the same compositional answer data.

tion tasks require writing a sentence with two provided skills sampled from the **literary** and **rhetorical** categories. Details can be found in Appendix D.

Evaluation Metrics. For Skill-Mix tasks, we use quality measure metrics for the generated sentence from Yu et al. (2024) (namely, *Full Marks* and *Skill Fraction*) based on a rubric, and use GPT-4o-mini as a judge. Details can be found in Appendix D.2. All other tasks are evaluated using *exact match* accuracy with a regex-based answer extractor to extract model answers from responses.

Zero-shot/Few-shot Baselines. Figure 3 summarizes the high-order variables of the configurations we evaluate. For zero-shot compositional generalization, we include the following baselines: (1) Few-shot direct answer prompting; (2) Few-shot CoT prompting; (3) Model merging of atomic CoT models (*StandardCoT-Merge*); (4) Multitask learning of atomic CoTs (*StandardCoT-MTL*).

Baselines with Compositional Supervision. We compare rejection sampling fine-tuning of Composable CoT models with the following baselines. (1) Continued fine-tuning (CFT) the multitask model of atomic CoTs (*CFT on StandardCoT-MTL*); (2) Continued fine-tuning the merged model of atomic CoTs (*CFT on StandardCoT-Merge*); (3)

Multitask learning of atomic CoTs and compositional answers (*StandardCoT + Comp Answer*).

Details of each baseline above can be found in Appendix E. The differences of methods we evaluate for each setting are summarized in Table 15.

Data Construction. For two-way compositions, we sample uniformly from 2 chain-of-thought tags, *<prefix>* and *<suffix>*, for data construction. At inference time, we first append *<prefix>* to the prompt and sample from the combined model. Then, we append *<suffix>* to the end of the generated response, and continue generation.

Models and Training. We use Llama 2 7B-base (Touvron et al., 2023) and Qwen2.5 7B-base (Yang et al., 2025) as models, and use LoRA (Hu et al., 2022) for supervised fine-tuning. Configuration details and hyperparameters are in Appendix H.

5 Results

5.1 Zero-shot Generalization

We evaluate the compositional generalization of ComposableCoT-Merge and ComposableCoT-MTL *without access to compositional supervision*. For all methods that we compare with, we control the amount of training data to be the same. Details of the training data can be found in Appendix J.

	String Tasks		Skill-Mix
	EM	Full Mark	Skill Fraction
StandardCoT-Merge	61.3	13.1	42.7
ComposableCoT-Merge	63.1	19.2	54.1
StandardCoT-MTL	82.3	28.2	55.9
ComposableCoT-MTL	86.7	33.1	61.0

Table 3: Zero-shot three-way composition. Combining ComposableCoT models outperforms combining StandardCoT models on the composition of three tasks.

ComposableCoT achieves better zero-shot generalization. Table 1 shows that ComposableCoT variants outperform baselines on a range of tasks for both models. Combining atomic CoT models trained on Composable CoT is better than combining models trained on standard CoT across settings, indicating better “composability” at inference time. Appendix I shows results where Composable CoT models are trained on more skills than are needed for the target task; they outperform standard CoT models in this setting as well.

5.2 Limited Compositional Supervision

We evaluate Composable CoT models after being further improved with one iteration of RFT using the limited compositional supervision. We compare with multitask learning and continued fine-tuning baselines given the same compositional answer dataset $D_{\mathcal{T}_{(i,j)}}$ of size $N_{(i,j)} \leq 500$. Details of the data condition are in Appendix J.

Table 2 shows that with the same compositional training data, **using RFT on top of ComposableCoT-MTL and ComposableCoT-Merge achieves the best compositional task performance**, outperforming multitask learning and continued fine-tuning baselines across settings.

Moreover, **RFT upon ComposableCoT models is generally better than RFT on StandardCoT models**. Using the same combination method (MTL or Model Merging), RFT upon ComposableCoT models outperforms the StandardCoT counterpart across models and tasks.

6 Results on Complex Compositions

6.1 Three-way Composition

We evaluate Composable CoT on zero-shot compositions of *three* atomic tasks on Qwen2.5-7B using the following compositional tasks: (1) Letter Concat + Next Letter + Mult (*String Tasks*); (2) Skill-Mix Literary + Rhetorical + Logical (*Skill-Mix*):

	Few-shot	StandardCoT	ComposableCoT
<i>Performance by Academic Subject</i>			
Algebra	60.7	60.7	65.0
Counting	38.4	41.8	44.1
Geometry	30.5	37.0	39.7
Inter. Algebra	21.9	23.8	28.3
Num. Theory	38.0	41.3	41.1
Prealgebra	62.5	66.0	65.4
Precalculus	21.1	22.0	20.0
<i>Overall</i>	42.2	44.6	46.5
<i>Performance by # of Required Skills</i>			
$k = 2$	56.0	58.2	59.1
$k = 3$	43.0	44.7	47.3
$k = 4$	32.0	34.5	37.4
$k = 5$	23.0	24.0	23.4

Table 4: Skill composition of ComposableCoT on MATH with Qwen2.5-7B. When trained on atomic skills for math reasoning, ComposableCoT models outperform StandardCoT models by 1.9% on average.

Write a sentence using skills from each of the three Skill-Mix categories. We compare ComposableCoT models with StandardCoT models constructed by model merging or multi-task learning. Details are in Appendix K.

Table 3 shows that combining ComposableCoT models performs better on three-way composition: for example, ComposableCoT-MTL outperforms StandardCoT-MTL by 4.8% on average.

6.2 Skill Composition for Math Reasoning

We evaluate ComposableCoT in a less controlled setting of math reasoning. We explore reasoning in the context of the MATH dataset (Hendrycks et al., 2021), where prior work shows that knowledge of different math subjects (skills in these subjects, e.g., “solving equations”) must be applied to solve problems (Didolkar et al., 2024; He et al., 2025).

Setup. We use the STAT dataset from He et al. (2025) where each example is a math word problem annotated with a CoT trace and one subject-specific (e.g., algebra) math skill required to solve it. For each subject, we select the five most frequently used skills in the training set as the atomic skills, and take the corresponding training examples to form atomic CoT data. Note that each example is created to specifically tailor one skill, but this setting does not necessarily feature a clear separation of skills like our string manipulation tasks do; see further discussions in Appendix M. Then, we fine-tune an atomic CoT model using Composable CoT format *for each subject* with the same scheme

	CoT	Perf.	% $\mathcal{T}_1$	% $\mathcal{T}_2$	% Both
Next Letter	Standard	3.6	0.0	100.0	0.0
+ Mult	Composable	96.3	98.9	100.0	[†] 98.9
Concat	Standard	60.9	100.0	66.7	66.7
+ Next Letter	Composable	63.3	100.0	85.9	[†] 85.9
Concat	Standard	72.1	99.7	32.1	32.1
+ Mult	Composable	74.3	100.0	83.1	[†] 81.3
Literary	Standard	42.0	65.3	58.0	37.6
+ Rhetorical	Composable	49.0	64.5	65.7	[†] 42.0

Table 5: Quality of CoTs generated by ComposableCoT zero-shot. “% $\mathcal{T}_1$ ” denotes the percentage of generated responses that use the CoT format of the first atomic task of the composition, and likewise for the second. [†] denotes that the ComposableCoT method has a significantly higher “% Both” than the StandardCoT counterpart at the 0.01 level using a paired bootstrap test. “Perf.” denotes the task performance.

described in Section 4. Finally, we evaluate the fine-tuned model on the test set of the corresponding subject in the MATH dataset, where each test example is considered a composition of atomic skills within that subject.

Training and Evaluation. We use Qwen2.5-7B-Base as the model and use multi-task learning as the combination method. We use the same low-data setting from Section 4 with 100 – 500 training examples for each subject. We compare against few-shot CoT prompting and SFT on StandardCoT data, and we use exact match accuracy as the evaluation metric. More details are in Appendix M.

Results. Table 4 shows that **combining ComposableCoT models outperforms combining StandardCoT models in multiple subjects in MATH**, with an average improvement of 4.3% over the base model and 1.9% over the StandardCoT model. We also show a breakdown by the number of required skills. ComposableCoT outperforms StandardCoT not only on $k = 2$, but also on $k = 3$ and $k = 4$ by substantial margins.

7 Analysis

We evaluate the intrinsic quality of model outputs by analyzing the occurrence of atomic CoT patterns in CoTs generated by ComposableCoT models for zero-shot composition. For the string manipulation and arithmetic tasks, we extract template-based patterns of each atomic CoT from the generated outputs of models evaluated on the compositional task. For Skill-Mix, we consider the CoT pattern of an atomic task to be used if the generated response

explicitly mentions the skill corresponding to that atomic skill category.

Table 5 shows results with models trained from Qwen 2.5-7B and combined with MTL. Details and more results are in Appendix L. **Combining ComposableCoT leads to higher presence of both atomic CoT patterns in the generated responses compared to StandardCoT.** Models trained with the Composable CoT use the combination of learned skills in some form more frequently than StandardCoT. Examples of generated CoTs are in Appendix N.

8 Related Work

Compositional generalization has been considered a core capability for human-level reasoning models (Piantadosi and Aslin, 2016; Werchan et al., 2015; Fodor and Pylyshyn, 1988; Lake and Baroni, 2023), and has been considered difficult for LLMs because of the limitations of the transformer architecture and autoregressive generation (Thomm et al., 2024; Dziri et al., 2023). Recent theoretical analyses show that the compositional reasoning capability of LLMs can be improved by generating CoT (Li et al., 2024, 2023), but empirical results show that non-trivial effort needs to be put through prompt engineering (Chen et al., 2024; Gao et al., 2024) or data selection (Khot et al., 2023; Zhou et al., 2023; Levy et al., 2023; Ye et al., 2023) to observe such improvements with CoT (Sprague et al., 2025), particularly in domains where compositional solutions to problems are crucial (Ding et al., 2024; Ye et al., 2025). Prior work has explored more principled approaches, but they usually rely on heuristics to determine data quality (Hase et al., 2024; Bengio et al., 2009) or involve computationally intensive methods (Sun et al., 2024; Conklin et al., 2021). Our method is inspired by methods for merging models of different capabilities (Wu et al., 2025; Ma et al., 2025; Tam et al., 2024), and our work is the first to use model merging for compositional generalization with CoT.

9 Conclusion

We propose Composable Chain-of-Thought, a data augmentation scheme to convert CoT data of atomic reasoning skills into a format that facilitates inference-time compositional generalization. Training atomic CoT models with Composable CoT and combining them with model merging or multitask learning leads to better zero-shot compositional rea-

soning performance than building models with the standard CoT format. Such a combined model can be further improved by a limited amount of compositional data with rejection sampling fine-tuning. Learning to reason with Composable CoT shows a promising approach to improve compositional reasoning in LLMs, and could be extended to build more efficient and robust large reasoning models.

Limitations

Our work focuses on the *systematicity* aspect of compositional generalization, the capability of applying known components in unseen combinations (Fodor and Pylyshyn, 1988). We consider the productivity and primitive application aspects, including length generalization (Anil et al., 2022; Zhou et al., 2024), out of scope for this work. We believe that the composable CoT technique could help in this setting, but we leave more rigorous development and experimentation for future work.

Furthermore, our experiments leverage small-scale datasets where both the atomic skills and compositional skills can be learned with a small amount of training data. Our focus here is on conducting controlled experiments, but we believe our methods and their principles can be scaled up to more complex settings, as empirical results on math skill compositions have demonstrated.

Finally, our exploration is limited to open-source language models operating on English-language datasets, due to the availability of models and data for the task settings at hand. As additional models and datasets become available, we believe there would be value in scaling this study further to understand the results in a wider range of settings.

Ethical Considerations

This work does not involve human subjects or the release of sensitive data. We do not clearly see harms arising from the applications of the proposed method. Therefore, we are not aware of any obvious ethical concern related to this work, except for those inherited from the development of large language models and generative AI technology more broadly.

Acknowledgments

Thanks to Katrin Erk, Kyle Mahowald, and other members of the TAUR lab for helpful discussion and suggestions. This work was partially supported by the Sloan Foundation, a grant from Open

Philanthropy, NSF CAREER Award IIS-2145280, the NSF AI Institute for Foundations of Machine Learning (IFML), and the NSF under Cooperative Agreement 2421782 and the Simons Foundation grant MPS-AI-00010515 awarded to the NSF-Simons AI Institute for Cosmic Origins — CosmicAI, <https://www.cosmicai.org/>. This research has been supported by computing support on the Vista GPU Cluster through the Center for Generative AI (CGAI) and the Texas Advanced Computing Center (TACC) at the University of Texas at Austin.

References

- Natalie Abreu, Edwin Zhang, Eran Malach, and Naomi Saphra. 2025. [A taxonomy of transcendence](#). In *Second Conference on Language Modeling*.
- Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, and 3 others. 2025. [Smollm2: When smol goes big – data-centric training of a small language model](#). *Preprint*, arXiv:2502.02737.
- Cem Anil, Yuhuai Wu, Anders Johan Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Venkatesh Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. 2022. [Exploring length generalization in large language models](#). In *Advances in Neural Information Processing Systems*.
- Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. [Curriculum learning](#). In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09*, page 41–48, New York, NY, USA. Association for Computing Machinery.
- 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.
- Karl Cobbe, Vineet Kosaraju, Mo 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*, abs/2110.14168.
- Henry Conklin, Bailin Wang, Kenny Smith, and Ivan Titov. 2021. [Meta-learning to compositionally generalize](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*

- Brenden M. Lake and Marco Baroni. 2023. [Human-like systematic generalization through a meta-learning neural network](#). *Nature*, 623:115 – 121.
- Itay Levy, Ben Bogin, and Jonathan Berant. 2023. Diverse demonstrations improve in-context compositional generalization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Toronto, Canada.
- Yingcong Li, Kartik Sreenivasan, Angeliki Giannou, Dimitris Papailiopoulos, and Samet Oymak. 2023. [Dissecting chain-of-thought: Compositionality through in-context filtering and learning](#). In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. 2024. [Chain of thought empowers transformers to solve inherently serial problems](#). In *The Twelfth International Conference on Learning Representations*.
- Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025. CoT-Valve: Length-Compressible Chain-of-Thought Tuning. *arXiv preprint arXiv:2502.09601*.
- Ryan Marten, Trung Vu, Charlie Cheng-Jie Ji, Kartik Sharma, Shreyas Pimpalgaonkar, Alex Dimakis, and Maheswaran Sathiamoorthy. 2025. Curator: A Tool for Synthetic Data Creation.
- Michael S Matena and Colin Raffel. 2022. [Merging Models with Fisher-Weighted Averaging](#). In *Advances in Neural Information Processing Systems*.
- 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*.
- OpenAI, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Ćwady, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, and 400 others. 2024. GPT-4o System Card. *arXiv preprint arXiv:2410.21276*.
- Steven Piantadosi and Richard Aslin. 2016. Compositional reasoning in early childhood. In *PloS one*, volume 11.
- QwenTeam. 2025. [QwQ-32B: Embracing the Power of Reinforcement Learning](#).
- Parshin Shojaee, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. 2025. [The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity](#).
- Zayne Rea Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. 2025. [To CoT or not to CoT? Chain-of-thought helps mainly on math and symbolic reasoning](#). In *The Thirteenth International Conference on Learning Representations*.
- Yiyu Sun, Shawn Hu, Georgia Zhou, Ken Zheng, Hannaneh Hajishirzi, Nouha Dziri, and Dawn Song. 2025. Omega: Can llms reason outside the box in math? evaluating exploratory, compositional, and transformative generalization. *arXiv preprint arXiv:2506.18880*.
- Zhiqing Sun, Longhui Yu, Yikang Shen, Weiyang Liu, Yiming Yang, Sean Welleck, and Chuang Gan. 2024. [Easy-to-hard generalization: Scalable alignment beyond human supervision](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Derek Tam, Yash Kant, Brian Lester, Igor Gilitschenski, and Colin Raffel. 2024. Realistic evaluation of model merging for compositional generalization. *arXiv preprint arXiv:2409.18314*.
- Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.
- Jonathan Thimm, Giacomo Camposampiero, Aleksandar Terzic, Michael Hersche, Bernhard Schölkopf, and Abbas Rahimi. 2024. [Limits of transformer language models on learning to compose algorithms](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- 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*.
- Denise M. Werchan, Anne G.E. Collins, Michael Joshua Frank, and Dima Amso. 2015. [8-month-old infants spontaneously learn and generalize hierarchical rules](#). *Psychological Science*, 26:805 – 815.
- Han Wu, Yuxuan Yao, Shuqi Liu, Zehua Liu, Xiaojin Fu, Xiongwei Han, Xing Li, Hui-Ling Zhen, Tao Zhong, and Mingxuan Yuan. 2025. Unlocking efficient long-to-short llm reasoning with model merging. *arXiv preprint arXiv:2503.20641*.

- Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2023. [TIES-merging: Resolving interference when merging models](#). In *Thirty-seventh Conference on Neural Information Processing Systems*.
- 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. 2025. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.
- Xi Ye and Greg Durrett. 2022. The unreliability of explanations in few-shot prompting for textual reasoning. In *Proceedings of NeurIPS*.
- Xi Ye, Srinivasan Iyer, Asli Celikyilmaz, Veselin Stoyanov, Greg Durrett, and Ramakanth Pasunuru. 2023. Complementary explanations for effective in-context learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, Toronto, Canada. Association for Computational Linguistics.
- Xi Ye, Fangcong Yin, Yinghui He, Joie Zhang, Howard Yen, Tianyu Gao, Greg Durrett, and Danqi Chen. 2025. LongProc: Benchmarking Long-Context Language Models on Long Procedural Generation. *arXiv preprint arXiv:2501.05414*.
- 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*.
- Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2024. [Scaling relationship on learning mathematical reasoning with large language models](#).
- Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. [STaR: Bootstrapping Reasoning With Reasoning](#). In *Advances in Neural Information Processing Systems*.
- 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, and Zheyang Luo. 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)*, pages 400–410, Bangkok, Thailand. Association for Computational Linguistics.
- Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 2023. [Least-to-most prompting enables complex reasoning in large language models](#). In *The Eleventh International Conference on Learning Representations*.
- Yongchao Zhou, Uri Alon, Xinyun Chen, Xuezhi Wang, Rishabh Agarwal, and Denny Zhou. 2024. [Transformers can achieve length generalization but not robustly](#). In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*.

A A Note on Composing Tasks

There exist various possible ways to combine atomic tasks into a compositional task with the combination function g . We simplify g into two types: (1) composite: the output of one atomic task is used as part of the input of another task, $g(\mathcal{T}_i, \mathcal{T}_j) = \mathcal{T}_i \circ \mathcal{T}_j$ or $g(\mathcal{T}_i, \mathcal{T}_j) = \mathcal{T}_j \circ \mathcal{T}_i$; (2) concatenation: the outputs of the two atomic tasks are concatenated using the same input, $g(\mathcal{T}_i, \mathcal{T}_j) = \mathcal{T}_i \oplus \mathcal{T}_j$ or $g(\mathcal{T}_i, \mathcal{T}_j) = \mathcal{T}_j \oplus \mathcal{T}_i$. Among tasks evaluated in Section 4, the string manipulation and arithmetic tasks need to be solved by a composite function, while the Skill-Mix task can be solved by either a composite function or a concatenation function.

B Design Choices for Proxy Prefixes

When designing the proxy prefix CoT, we would like to consider the following constraints. (1) We do not assume any prior knowledge about what would possibly be put in the proxy prefix at inference time; (2) We do not assume strong relevance between the proxy prefix CoT and the actual CoT, i.e., not all the information in the proxy prefix CoT is useful for predicting the CoT and the final answer. Otherwise, the models might learn to copy the prefix CoT verbatim.

B.1 Ablation of Proxy Prefix Variants on Compositions

Based on the considerations above, we experiment with the following variants on the compositional setting:

- **Random Letters:** We uniformly sample random, lower-case letters from the alphabet to form a sequence to simulate an *arbitrary* prefix CoT. For each training example with a CoT trace of m letters, the length of the prefix sequence is randomly drawn from a uniform distribution between $0.5m$ and $1.5m$, so that the random sequences have a similar length distribution as the atomic CoT traces.

- **Scrambled CoT:** Given the ground truth CoT in an atomic CoT training example, we randomly shuffle the words in the ground truth CoT, combine them again into a sequence in a random order, and use the combined sequence as the prefix CoT. The tokens are in-distribution for the reasoning task, but the random ordering removes the structure typical of CoTs. Note that the ground truth CoT then appears twice: once scrambled, in the prefix, and again as the training target.
- **Random Text:** We sample random sentences from OpenWebText (Gokaslan and Cohen, 2019) and SmoLLM (Allal et al., 2025) to give prefix CoTs the same distribution as pre-training data.
- **Random Text plus CoT:** We sample random sentences from the pre-training corpora and sentences from the ground truth CoT in the atomic CoT training example. We then concatenate these sentences in random order as the prefix CoT. Note that in this case, the model can simply copy from the proxy prefix to generate the target atomic reasoning.
- **Random In-distribution (ID) CoT:** We randomly sample a ground truth atomic CoT from a different training example of the *same* atomic task, and use it as the prefix. This removes the model’s ability to copy the prefix CoT verbatim to do reasoning, while providing in-distribution CoT style and tokens.

We fine-tune Qwen2.5-7B on Composable CoT data for the string manipulation and arithmetic tasks using the prefix variants above. Then, we evaluate the fine-tuned models on zero-shot two-way compositions. Table 6 shows that using random letters as the prefixes leads to the best zero-shot compositional generalization among all variants. Interestingly, we note that the average performance generally decreases when the prefix distribution gets more similar to the atomic CoT distribution: while both use randomized sequences as prefixes, scrambled CoT underperforms random letters because scrambled CoT uses in-distribution words; while both use irrelevant sentences as prefixes, random ID CoT underperforms random text because random ID CoT uses in-distribution sentences. We hypothesize that this is because using in-distribution prefixes encourages models to copy the prefix CoT

into the suffix CoT rather than generating a different CoT as the continuation, leading to less robust inference-time behaviors.

B.2 Ablation of Proxy Prefix Variants on Atomic Tasks

To further investigate the impact of different proxy prefix variants, we evaluate the following proxy prefix variants on atomic tasks, particularly to test the behavior under distribution shifts. We mainly investigate **random letters**, **random text**, and **random text from the prompt**, where we sample random letters and words from the prompt $\mathbf{q}$ to form a sequence of random lengths, to simulate three different distributions of prefix CoT.

We evaluate these variants by fine-tuning models on Composable CoT datasets **that only use the following augmentation:** $d' = \mathbf{q} \langle \text{prefix} \rangle [\text{proxy prefixes}] \langle / \text{prefix} \rangle \langle \text{suffix} \rangle \text{ta} \langle / \text{suffix} \rangle$. Note that this is different from the setting discussed in Section 3.1 where the Composable CoT dataset consists of other possible augmentations as well based on the sampling of the tags (e.g., $d' = \mathbf{q} \langle \text{prefix} \rangle \text{ta} \langle / \text{prefix} \rangle$ when $k = 1$). This experiment mainly aims at stress testing the model’s capability of learning a single atomic task with a given proxy prefix CoT variant. We use the same hyperparameter configurations for all proxy prefix variants for a given task.

We evaluate the fine-tuned models on the in-domain task in two settings: (1) *In-domain prefix*: we append the same type of prefix as we have used for training to the end of the prompt of the in-domain test example and evaluate the model on it; (2) *Out-of-domain prefix*: we randomly sample a prefix from the other two variants and append it to the end of the prompt of the in-domain test example and evaluate the model on it. We run experiments on the three string manipulation and arithmetic tasks and report the average performance. Table 7 shows that although using random letters as the proxy prefix leads to the worst in-domain performance, it generalizes the best to out-of-domain prefixes, which is a more desirable behavior.

C Details of String Manipulation and Arithmetic Tasks

C.1 Task Descriptions

We consider the following atomic tasks.

- (1) **Next Letter in Alphabet:** We adapt this task from Efrat et al. (2023); Edman et al. (2024).

	Next Letter + Mult	Concat + Next Letter	Concat + Mult	Avg.
Random Letters	96.3	63.3	74.3	78.0
Scrambled CoT	93.3	59.3	73.6	75.4
Random Text plus CoT	63.3	61.5	57.0	60.6
Random Text	63.3	58.7	40.4	54.1
Random ID CoT	65.4	48.0	7.6	40.3

Table 6: Zero-shot **composition** performance of Composable CoT models fine-tuned on different variants of proxy prefix on Qwen2.5-7B. Using random letters as the proxy prefix achieves the best performance.

Type of Proxy Prefix	Exact Match Accuracy	
	In Domain Prefix	Out-of-Domain Prefix
Random Letters	83.0	90.0
Random Text from the Prompt	86.4	82.5
Random Text	90.6	70.0

Table 7: Atomic task performance of **atomic** CoT models fine-tuned on different variants of proxy prefix on Llama 2-7B. Using random letters as the proxy prefix achieves the best performance when evaluated with an unseen prefix at inference time.

Given a sequence of letters, find the next letter in the alphabet following the last letter.

(2) **Letter Concatenation:** We adapt this task from Wei et al. (2022); Zhou et al. (2023). Given a sequence of words, concatenate the first, second, last, or second-to-last letter of each word.

(3) **ASCII Multiplication:** We adapt this task from multi-digit multiplication tasks in Dziri et al. (2023); Gambardella et al. (2024). Given a sequence of words and a letter, find the ASCII value of the given letter multiplied by a given constant value.

We consider the following pairwise composition of the atomic tasks.

(1) **Next Letter + Multiplication:** Given a sequence of letters, find the next letter in the alphabet following the last letter, determine its ASCII value, and then perform multiplication with a given constant.

(2) **Concatenation + Next Letter:** Given a sequence of words, concatenate the first, second, or second-to-last letter of each word and then find the next letter in the alphabet following the last letter of the concatenated sequence.

(3) **Concatenation + Multiplication:** Given a sequence of words, concatenate the first, second, or second-to-last letter of each word, find the ASCII value of the last letter of the concatenated sequence, and then perform multiplication.

C.2 Data Construction

Next Letter in Alphabet We synthetically generate data for next letter in alphabet. We randomly sample letters from the English alphabet of a random length and concatenate them into a sequence. Then we extract the last letter from the sequence and derive the next letter following it in the alphabet. An example can be found in Example C.2. We automatically generate a chain-of-thought for each generated problem, using a fixed template shown in Example C.2.

ASCII Multiplication Similarly, we randomly sample letters from the English alphabet of a random length and concatenate them into a sequence. Then, we randomly sample another letter s and randomly sample an integer $a \in \{1, \dots, 9\}$. We find the ASCII value of s as $f(s)$ and compute the product $af(s)$ as the gold answer. An example can be found in Example C.3. We automatically generate a chain-of-thought for each generated problem, using a fixed template shown in Example C.3.

Letter Concatenation We follow Wei et al. (2022) to generate the dataset by randomly sampling from the most popular first and last names in the United States and the United Kingdom from <https://namecensus.com> and randomly concatenating them into a sequence of names. While the original task in Wei et al. (2022) only requires concatenating the last letter of each name together, we

raise the difficulty level by randomly asking for concatenations of the first, second, second-to-last, or the last letter. An example can be found in Example C.1. The CoT template is also shown in Example C.1.

Compositional Tasks We synthetically construct the compositional tasks of the string manipulation and arithmetic tasks in similar procedures as used to generate the atomic data. An example of next letter + ASCII multiplication can be found in Example C.4, concatenation + next letter in Example C.5, and concatenation + multiplication in Example C.6. We made a design decision to exclude one variant of concatenation + next letter that concatenates the last letter of each word and finds the next letter following the last letter in the concatenated sequence; this variant can be solved by the reasoning shortcut of only applying next letter in alphabet rather than a composition of both.

C.1 Atomic Task Example: Letter Concatenation Example

[Instruction]

Take the second-to-the-last letter of each word in the sequence and concatenate them in lower case: Tequan Monjur Khia Jodi-leigh answer

[Chain-of-Thought + Answer String]

The second-to-the-last letter of the 1st word is a. The second-to-the-last letter of the 2nd word is u. The second-to-the-last letter of the 3rd word is i. The second-to-the-last letter of the 4th word is g. So the answer is auig.

[Answer String]

auig

C.2 Atomic Task Example: Next letter in alphabet

[Instruction]

Find the Next letter in alphabet following the last letter in the sequence: wqsibn nicdlpwqbnoicdcxcxrfoilpcbnixuc bssssejxuzods answer:

[Chain-of-Thought + Answer String]

The last letter is s, and the letter following it in alphabet is t. So the answer is t.

[Answer String]

t

C.3 Atomic Task Example: ASCII Multiplication

[Instruction]

Find the ASCII value of the letter after '<letter>' and multiply the ASCII value by 2: byaxaxcpoteznwnwseselyjlrctx txcbfvmfezbyplymfotjbfbvj lhotzjbjcpcbtzhorepyjckofj <letter> d answer:

[Chain-of-Thought + Answer String]

The ASCII value of the letter d is 100, and multiplying the ASCII value by 2 gives us 200. So the answer is 200.

[Answer String]

200

C.4 Compositional Task Example: Next letter + ASCII Multiplication

[Instruction]

Find the ASCII value of the Next letter in alphabet following the last letter in the sequence and multiply the ASCII value by 5: knnxqsxvshqugxfuquljumsbihgxvqihnxuufuknxvumuupkpkshljqsbkiz answer:

[Answer String]

485

C.5 Compositional Task Example: Concatenation + Next Letter

[Instruction]

Take the second-to-the-last letter of each word in the sequence, concatenate them in lower case, and find the Next letter in alphabet following the last letter in the sequence of the concatenated letters: Tyjai Ahijah Denzil Amine answer:

[Answer String]

o

C.6 Compositional Task Example: Concatenation + Multiplication

[Instruction]

Take the second-to-the-last letter of each word in the sequence, concatenate them in lower case, then find the ASCII value of the last letter in the sequence of the concatenated letters, and multiply the ASCII value by 3: Zarriah Amylee Li Javarie answer:

[Answer String]

315

D Details of Skill-Mix Tasks

D.1 Modifications of Skill-Mix

We adapt the Skill-Mix dataset from Yu et al. (2024). For each example, the model is given a natural language skill, its definition, an example of the skill, and a topic to focus on, and the model needs to write a grammatical sentence to demonstrate the skill on the topic. Because we mainly focus on pairwise composition, we only use the $k = 2$ and $k = 1$ composition sets of the Skill-Mix data. We apply the following modifications to the dataset to fit our setting of compositional reasoning.

1. Filtering the categories of skills: We keep examples with skills of the rhetorical and literary categories out of the five categories from the original dataset. This is because the rhetorical and literary skills have the least overlap while other categories have more (e.g. the logical and rhetorical skills have a large body of overlaps).
2. Removing the requirements of post-hoc explanation and refinement from the prompt. The original dataset evaluates models by prompting the models to first write a sentence, provide an explanation for the written sentence, and then do another round of refinement based on feedback from the grader (an LLM-as-a-judge). To fit into our setting of chain-of-thought reasoning and direct answering, we remove these irrelevant elements in the prompt.

3. Using a public test set: The original evaluation of Yu et al. (2024) hides the test set from the public and models can only be evaluated based on API calls to the hidden test set. To ensure reproducibility of our results, we use a public test set collected by Zhao et al. (2024).

As an open-ended generation task, Skill-Mix does not have a single ground truth sentence. Zhao et al. (2024) collect a silver-labeled dataset of sentences and their post-hoc explanations generated by GPT-4o (OpenAI et al., 2024) and graded as correct by the grader. We further prompt GPT-4o to transform the (sentence, explanation) pairs into (CoT, sentence) pairs to resolve potential coreference issues. Examples of the prompt, CoT traces, and sentence can be found in Examples D.1, D.2, and D.3.

D.1 Atomic Task Example: Skill-Mix-Literary

[Instruction]

Greetings! I am interested in natural language processing and I was wondering if you could help me generate an example of text that illustrates multiple skills in semantics or syntax. The example should be a minimal natural piece of text with up to a few lines in the context of Vikings that illustrates all of the following skills: oxymoron. Please keep the text as short as possible, and make sure the concepts can be found fully from the text.

For reference, here are the definitions and examples for the concepts:

****oxymoron****: A figure of speech in which apparently contradictory terms appear in conjunction. For example, “The hall was filled with a deafening silence.” Here, the phrase “deafening silence” is an oxymoron.

Please start the minimal natural piece of text with ‘Answer:’.

Thanks very much!

[Chain-of-Thought + Answer String]

Explanation: The goal is to construct a sentence that showcases the use of

an oxymoron while remaining relevant to the Viking theme. To achieve this, we focus on creating a phrase that contains seemingly contradictory terms. We begin with the context of Vikings enduring harsh weather conditions. The word “fiery” conveys an intense heat, while “frost” indicates severe cold. When these terms are combined into “fiery frost,” they form an oxymoron as they depict both extreme temperature sensations simultaneously. This juxtaposition effectively creates a semantic paradox, leading to vivid imagery and emphasizing the harsh conditions faced by Vikings.

Answer: “In the bitter cold, the Viking felt a fiery frost.”

[Answer String]

Answer: “In the bitter cold, the Viking felt a fiery frost.”

D.2 Atomic Task Example: Skill-Mix-Rhetorical

[Instruction]

Greetings! I am interested in natural language processing and I was wondering if you could help me generate an example of text that illustrates multiple skills in semantics or syntax. The example should be a minimal natural piece of text with up to a few lines in the context of Hiking that illustrates all of the following skills: begging the question or assuming the conclusion. Please keep the text as short as possible, and make sure the concepts can be found fully from the text.

For reference, here are the definitions and examples for the concepts:

****begging the question or assuming the conclusion****: A circular argument that paraphrases the question. For example, “Of course the earth is round; people have known this for hundreds of years.”

Please start the minimal natural piece of text with ‘Answer:’.

Thanks very much!

[Chain-of-Thought + Answer String]

Explanation: To illustrate the concept of begging the question or assuming the conclusion within the context of hiking, we start by identifying a statement that relies on its own conclusion as a premise. First, recognize that the claim “hiking is beneficial” needs supporting evidence to be considered valid. However, if we use the phrase “because it’s good for your health” as justification, we notice that it merely reiterates the initial claim using slightly different wording. This is because declaring something “beneficial” inherently implies a positive impact, such as being “good for your health.” Thus, the reasoning becomes circular, as it depends on the same assumption it seeks to prove.

Answer: “Hiking is beneficial because it’s good for your health.”

[Answer String]

Answer: “Hiking is beneficial because it’s good for your health.”

D.3 Compositional Task Example: Skill-Mix Literary + Rhetorical

[Instruction]

Greetings! I am interested in natural language processing and I was wondering if you could help me generate an example of text that illustrates multiple skills in semantics or syntax. The example should be a minimal natural piece of text with up to a few lines in the context of Vikings that illustrates all of the following skills: anaphora resolution, begging the question or assuming the conclusion. Please keep the text as short as possible, and make sure the concepts can be found

fully from the text.

For reference, here are the definitions and examples for the concepts:

****anaphora resolution****: Resolving the antecedent of a pronoun or noun phrase. For example, “The car is falling apart, but it still works.” Here, “it” is the anaphor and “car” is the antecedent.

*****begging the question or assuming the conclusion****: A circular argument that paraphrases the question. For example, “Of course the earth is round; people have known this for hundreds of years.”

Please start the minimal natural piece of text with ‘Answer:’.

Thanks very much!

[Answer String]

Answer:

The Viking chief, undefeated thanks to his ship, asserted, “It remains unconquered because it’s the ‘Indomitable’.”

D.2 Evaluation Metrics

We use GPT-4o-mini as the LLM-as-a-judge to grade the generated sentence using the exact same grading rubric as provided by Yu et al. (2024); the grader judges the quality of the sentence based on if: (1) All skills are used; (2) The sentence makes sense; (3) The sentence attaches to the given topic; (4) The sentence is short. We use the evaluation metrics for each generated sentence in Yu et al. (2024), including the following:

1. **Full Marks**: 1 if the generated sentence satisfies all four criteria above and 0 otherwise.
2. **Skill Fraction**: The fraction of skills being demonstrated if all the other three criteria are satisfied; 0 otherwise

We aggregate these metrics by averaging over all generated responses. In general, full marks evaluate the model’s capability of writing a perfect sentence for the task, while skill fraction evaluates how good the model is at handling skills given that it is good at the other stylistic capabilities. We use Curator (Marten et al., 2025) for an efficient implementation of the evaluation pipeline.

D.3 RFT on Skill-Mix Tasks with Rationalization

For open-ended generation tasks like Skill-Mix, it is hard to only use the reference answer to verify the correctness of the responses sampled from a model. Thus, we use rationalization to perform rejection sampling fine-tuning for Skill-Mix: we first append the direct answer label to the end of the prompt and sample post-hoc explanations for the given answer from the model; because M_{comb} is optimized to generate an answer following a CoT, we extract the generated answer following the generated explanation and filter out explanations whose following answer is not the same as the provided gold answer; finally, we use the accepted explanations as surrogates for CoT to form the RFT data.

E Details of Baselines

Zero-shot/Few-shot Baselines. We include the following baselines: (1) Few-shot direct answer prompting: we prompt M_0 with 5-shot demonstrations drawn from the compositional data; (2) Few-shot CoT prompting: we prompt M_0 with 5-shot CoT demonstrations drawn from the *atomic* data; (3) Model merging of atomic CoT models (*StandardCoT-Merge*): we fine-tune two models M_i and M_j based on M_0 with $D_{\mathcal{T}_i}^{\text{CoT}}$ and $D_{\mathcal{T}_j}^{\text{CoT}}$ respectively and merge them into M_{comb} with Task Arithmetic; (4) Multitask learning of atomic CoTs (*StandardCoT-MTL*): we fine-tune M_0 to be a single multitask learning model $M_{\text{SCoT-MTL}}$ on $D_{\mathcal{T}_i}^{\text{CoT}} + D_{\mathcal{T}_j}^{\text{CoT}}$.

Baselines with Compositional Supervision.

We use the following baselines: (1) Continued fine-tuning (CFT) the multitask model of atomic CoTs (*CFT on StandardCoT-MTL*): we continue fine-tune the multitask model $M_{\text{SCoT-MTL}}$ on $D_{\mathcal{T}_{(i,j)}}$; (2) Continued fine-tuning the merged model of atomic CoTs (*CFT on StandardCoT-Merge*): we continue fine-tune the merged model of the two atomic CoT models M_{comb} on $D_{\mathcal{T}_{(i,j)}}$; (3) Multitask learning of atomic CoTs and compositional answers (*StandardCoT + Comp Answer*): we fine-tune a single multitask learning model based on M_0 on the combined dataset of $D_{\mathcal{T}_i}^{\text{CoT}} + D_{\mathcal{T}_j}^{\text{CoT}} + D_{\mathcal{T}_{(i,j)}}$. We also include supervised learning baselines (SFT) where M_0 is fine-tuned on the same compositional answer data $D_{\mathcal{T}_{(i,j)}}$.

F Single-Task Learning Performance

We report the single-task learning performance of the atomic CoT models by evaluating them on the in-domain atomic tasks. We would like the atomic tasks to be easy to learn to reflect the practical settings where we train models on basic, easy-to-learn skills and generalize to harder, unseen tasks. The training data conditions and hyperparameters for training can be found in Appendix H. Table 8 shows that all atomic tasks we evaluate are learnable within a small amount of training data ($N_i, N_j \leq 500$).

In addition, we observe that training on ComposableCoT or StandardCoT does not lead to consistent differences in atomic CoT performance, while the exception is on Skill-Mix-Rhetorical for Llama 2-7B where fine-tuning on ComposableCoT outperforms fine-tuning on StandardCoT by a large margin.

G Base Model Performance on Evaluation Tasks

To confirm that our task design includes evaluation tasks that are less commonly seen in the pretraining data of LLMs, we evaluate the perplexity of the task datasets.

We compare the datasets of the string manipulation and arithmetic datasets used in our experiments with mathematical reasoning data (GSM8K (Cobbe et al., 2021)), and instruction following data (Alpaca (Taori et al., 2023)) in terms of perplexity: we compute the average perplexity score of pre-trained LLMs over the concatenation of the question and ground-truth chain-of-thought response to examine how predictable the task is; the lower the perplexity is, the more predictable and the easier the task is to learn. We also include the perplexity over the pretraining corpus as a reference point.

Table 9 indicates that our selected tasks consist of text that is less typical under pre-trained language models than other similar tasks, particularly other popular reasoning tasks. The higher perplexity is likely due to these tasks requiring the model to operate on letters rather than words.

H Training Configurations

H.1 General Training Configurations

We conduct all fine-tuning experiments with LoRA (Hu et al., 2022) using the following set of hyperparameters: we use a rank of 8, $\alpha = 16$, and

a dropout rate of 0.2 to prevent overfitting. We apply LoRA adapters to all linear modules, including the attention matrices Q , K , V and MLP matrices of all layers. We use bfloat16 precision for training and we use the efficient implementation of LoRA by LlamaFactory (Zheng et al., 2024). We use a training batch size of 4 and train for 5 epochs for all experiments that share the same number of training data; for methods that use a potentially smaller amount of training data (e.g. RFT methods usually get fewer data examples than the number of compositional training data provided, depending on how many correct responses we can sample from the model), we adjust the batch size to match the number of steps.

H.2 General Inference Configuration

For rejection sampling, we sample 10 responses for each prompt and use temperature $\tau = 0.9$ for inference; otherwise, we use greedy decoding.

H.3 Configuration for Rejection Sampling Fine-tuning

In addition to the sampling parameters (see Section 4), we consider the following configuration of RFT for sampling the correct responses: if the model generates multiple correct responses for a given question, we only randomly select *one* of them to be added into the RFT dataset D_{RFT} . In this way we ensure the diversity of examples in D_{RFT} so that the dataset will not be filled with samples from a small set of examples where the model is good at. For Skill-Mix tasks, we perform rationalization for RFT because it is open-ended generation (see Section 3.3).

H.4 Hyperparameters: Learning Rate

We find in preliminary experiments that learning rate is the most important hyperparameter for the fine-tuning experiments of our interest. We perform hyperparameter sweeps for learning rate over the space of $\{5e-3, 1e-3, 5e-4, 1e-4, 5e-5\}$ on a validation set for each experiment. The optimal learning rate for each method for the experiments with compositional supervision in Table 10.

H.5 Hyperparameters: Model Merging

For methods that use model merging as the combination, we use Task Arithmetic (Ilharco et al., 2023) to combine the atomic CoT models. For two-way compositions, we perform a hyperparameter sweep for the scalars α_1 over the space of

CoT Format	Next Letter EM	ASCII Mult EM	Concat EM	Skill-Mix Literary Full Marks	Skill-Mix Literary Skill Fraction	Skill-Mix Rhetorical Full Marks	Skill-Mix Rhetorical Skill Fraction
Llama 2-7B							
StandardCoT	100.0	85.7	83.0	63.5	63.5	53.3	53.3
ComposableCoT	95.0	86.0	77.0	71.4	71.4	72.4	72.4
Qwen 2.5-7B							
StandardCoT	90.0	99.0	77.4	77.6	77.6	70.5	70.5
ComposableCoT	99.4	99.7	77.3	77.6	77.6	76.7	76.7

Table 8: Single-task learning performance by evaluating the atomic CoT models on the in-domain atomic tasks.

	Task Dataset	Pretraining WikiText	Math Reasoning GSM8K	Instruction Following Alpaca	String Manipulation And Arithmetic Avg.
Model	Llama 2-7B	4.77	2.54	3.84	15.97
	Qwen 2.5-7B	5.93	2.38	4.68	7.02

Table 9: Perplexity of the base models over the task datasets. For the string manipulation and arithmetic tasks, the perplexity score is averaged over the 3 atomic tasks and the 3 pairwise compositional tasks. Our evaluation datasets include text that is less predictable under pre-trained LLMs than other similar tasks.

Category	Method	Next Letter + Mult	Concat + Next Letter	Concat + Mult	Skill-Mix Literary + Rhetorical
Llama 2-7B					
SFT	SFT on Base Model	1e-3	1e-3	5e-4	5e-4
	CFT on StandardCoT-Merge	1e-3	5e-4	1e-4	1e-4
	CFT on StandardCoT-MTL	1e-4	1e-4	1e-4	1e-3
MTL	StandardCoT + Comp Answer	1e-3	5e-4	1e-3	5e-4
RFT	StandardCoT-Merge	-	1e-3	1e-3	5e-4
	ComposableCoT-Merge (Ours)	1e-4	1e-4	1e-3	1e-3
Qwen 2.5-7B					
SFT	SFT on Base Model	-	1e-3	1e-3	5e-4
	CFT on StandardCoT-Merge	-	5e-4	5e-4	1e-4
	CFT on StandardCoT-MTL	-	1e-3	1e-3	1e-3
MTL	StandardCoT + Comp Answer	-	5e-4	5e-4	1e-3
RFT	StandardCoT-MTL	-	1e-3	1e-4	5e-4
	ComposableCoT-MTL (Ours)	-	1e-3	1e-3	5e-4

Table 10: Optimal learning rate for each method in the experiments with compositional supervision.

		Standard Composable	
Next Letter + Mult	EM	56.1	75.9
Concat + Next Letter	EM	39.1	46.2
Concat + Mult	EM	44.3	48.9
Skill-Mix Literary	Full Mark	37.1	42.0
+ Rhetorical	Skill Fraction	55.1	62.7

Table 11: Zero-shot generalization on two-way compositions when merging **three** atomic models (i.e., there is a distractor skill). Merging ComposableCoT models is better than merging StandardCoT models in this setting.

$\alpha_1 \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ on a validation set for each task. Another scaling factor is $\alpha_2 = 1 - \alpha_1$.

I Two-way Composition with Larger Skill Pools

In practice, models may need to have many capabilities to address problems of interest. Compared to our existing settings, such models need to select the skills to engage with for a particular task out of a larger pool of learned skills.

To evaluate this scenario, we train atomic models on *three* atomic skills on Qwen 2.5-7B and evaluate the combined model (with model merging) on the zero-shot composition of *two* of the atomic skills. This setup provides the model with additional potential combinations of skills to reason about. For the Skill-Mix tasks, we train on an additional atomic task, logical reasoning. Table 11 shows that learning ComposableCoT outperforms using StandardCoT by 8.8% on average, indicating that ComposableCoT models can select the appropriate skills to compose out of many learned skills.

J Data Statistics

J.1 General Data Conditions for Experiments

Table 12 summarizes the amount of training data and test data used in the evaluations in Sections 5.1 and 5.2. Note that for letter concatenation + multiplication we have two sizes of the compositional training data for Llama 2-7B and Qwen 2.5-7B: this is because all methods on Llama 2-7B perform poorly on zero-shot evaluation for this task and we need a slightly larger amount of compositional training data so that different methods can start to show distinguishable compositional task performance from each other. Regardless, we still consider 500 to be a reasonably small amount of training data, satisfying our ideal data conditions

defined earlier.

J.2 Training Data Used by Each Method

We show a detailed breakdown in Table 13 of the amount of training data used by each zero-shot method for both models and in Table 14 for Qwen 2.5-7B by each method with compositional answer data in the experiments in Section 5.2. Note that the statistics for Llama 2-7B in the setting with compositional supervision are mostly the same except $N_{(i,j)} = 500$ for concat + next letter and concat + mult.

K Details of Three-way Compositions

K.1 Data

We include 700 test examples for Letter Concat + Next Letter + Mult (*String Tasks*), and 245 test examples for Skill-Mix Literary + Rhetorical + Logical (*Skill-Mix*). For *Skill-Mix*, we additionally train an atomic model for Skill-Mix-Logical with 100 training examples.

K.2 Training and Inference

Training We use the same data augmentation scheme to create atomic CoT training data as the one we use for two-way composition in Section 4. This means that we append only one proxy prefix to the prompt. The general scheme can insert at most $n - 1$ proxy prefixes at the end of the prompt for $n > 2$, but we found that the test-time generalization scheme described in **Instantiation of Tags** under Section 3.1 works as well: adding only one proxy prefix achieves comparable compositional performance to adding two proxy prefixes while being more efficient during training, since the training data length is shorter. Thus, we experiment with the latter scheme.

Inference We use the same inference strategy specified in **Data Construction** under Section 4: for zero-shot inference, we first sample a response from M_{comb} . Then, we repeat the following *twice*: we append $\langle \text{suffix} \rangle$ to the end of the generated response when it stops generation, and continue generation until the model stops again.

L Full Results for the Quality Analysis of the Generated CoTs

Model Merging Results Table 16 includes the full results of the quality analysis of the generated CoTs using both multi-task learning (MTL) and

		# Train	# Test
Atomic Tasks	Next Letter	100	700
	ASCII Mult	100	700
	Concat	500	700
	Skill-Mix Literary	100	126
	Skill-Mix Rhetorical	100	105
Compositional Tasks	Next Letter + Mult	100	700
	Concat + Next Letter	100	504
	Concat + Mult (Llama 2-7B)	500	700
	Concat + Mult (Qwen 2.5-7B)	100	700
	Skill-Mix Literary + Rhetorical	100	245

Table 12: Data conditions for each task used for our evaluation.

	Method	N_i	N_j
Next Letter + Mult; Skill-Mix Literary + Rhetorical	StandardCoT-Merge	100	100
	ComposableCoT-Merge	100	100
	StandardCoT-MTL	100	100
	ComposableCoT-MTL	100	100
Concat + Next Letter; Concat + Mult	StandardCoT-Merge	500	100
	ComposableCoT-Merge	500	100
	StandardCoT-MTL	500	100
	ComposableCoT-MTL	500	100

Table 13: The detailed breakdown of the number of training data used by each method in the zero-shot setting. N_i and N_j denote the number of training data from the atomic tasks $\mathcal{T}_i$ and $\mathcal{T}_j$ seen by the method during training.

	Method	N_i	N_j	$N_{(i,j)}$
Next Letter + Mult; Skill-Mix Literary + Rhetorical	SFT on Base Model	0	0	100
	CFT on StandardCoT-Merge	100	100	100
	CFT on StandardCoT-MTL	100	100	100
	MTL on StandardCoT + Comp Answer	100	100	100
	RFT on StandardCoT-Merge	100	100	100
	RFT on ComposableCoT-Merge	100	100	100
	RFT on StandardCoT-MTL	100	100	100
	RFT on ComposableCoT-MTL	100	100	100
Concat + Next Letter; Concat + Mult	SFT on Base Model	0	0	100
	CFT on StandardCoT-Merge	500	100	100
	CFT on StandardCoT-MTL	500	100	100
	MTL on StandardCoT + Comp Answer	500	100	100
	RFT on StandardCoT-Merge	500	100	100
	RFT on ComposableCoT-Merge	500	100	100
	RFT on StandardCoT-MTL	500	100	100
	RFT on ComposableCoT-MTL	500	100	100

Table 14: The detailed breakdown of the number of training data used by each method with compositional supervision for Qwen 2.5-7B. N_i and N_j denotes the number of training data from the atomic tasks $\mathcal{T}_i$ and $\mathcal{T}_j$ seen by the method during training. $N_{(i,j)}$ denotes the number of compositional answer data seen during training.

Method	# Atomic CoT Models Trained	Atomic CoT Format	Combination Method	Model trained on Compositional Data	How is Compositional Data Used
<i>Zero-shot Evaluation</i>					
StandardCoT-Merge	2	Standard	Merge	-	-
ComposableCoT-Merge (Ours)	2	Composable	Merge	-	-
StandardCoT-MTL	1	Standard	MTL	-	-
ComposableCoT-MTL (Ours)	1	Composable	MTL	-	-
<i>Evaluation with Limited Compositional Answer Data</i>					
CFT on StandardCoT-Merge	2	Standard	Merge	StandardCoT-Merge	CFT
CFT on StandardCoT-MTL	1	Standard	MTL	StandardCoT-MTL	CFT
MTL on StandardCoT + Comp Answer	-	Standard	-	Pretrained Model	Mix with Atomic CoT data and MTL
RFT on StandardCoT-Merge	2	Standard	Merge	StandardCoT-Merge	RFT
RFT on ComposableCoT-Merge (Ours)	2	Composable	Merge	ComposableCoT-Merge	RFT
RFT on StandardCoT-MTL	1	Standard	MTL	StandardCoT-MTL	RFT
RFT on ComposableCoT-MTL (Ours)	1	Composable	MTL	ComposableCoT-MTL	RFT

Table 15: Summary of methods evaluated in the zero-shot compositional evaluation and the composition with limited compositional answer data. “Merge” stands for model merging; “MTL” stands for multitask learning; “CFT” stands for continued fine-tuning; “RFT” stands for rejection sampling fine-tuning. “-” means the property is not applicable to the method (e.g. *MTL on Standard + Comp Answer* mixes Standard CoT data with compositional answer data, and trains a single MTL model from the pretrained model, so there is no atomic CoT model trained or combined.)

model merging as the combination methods for atomic CoT models. Given the same combination method, combined Composable CoT models generate responses including both atomic CoT patterns more frequently than combined atomic CoT models.

M Details of Skill Composition for Math Reasoning

M.1 Setup

We follow He et al. (2025) to evaluate Composable-CoT for math reasoning. Models are fine-tuned on a training set of math word problems and their solutions in CoT format, each of which is designed to demonstrate the use of a specific skill for mathematical reasoning. Then, they are evaluated on the test set of the MATH dataset to solve math word problems. He et al. (2025) show that models fine-tuned with such skill-targeted data perform better on the MATH test set and other OOD math reasoning datasets than models fine-tuned on the vanilla MATH training set and other SFT datasets that are not designed to teach math reasoning skills. We use the training dataset *STAT-Syn* created by He et al. (2025) and formulate the setting as a skill composition problem.

Atomic tasks Following the definitions in Dikolkar et al. (2024), He et al. (2025) curate a list of mathematical skills for each math subject that

are useful for problem solving. We treat these as atomic reasoning skills for the purpose of this study. Each training example in *STAT-Syn* is designed to feature a single skill, making it ideally suited for our setting.

For each math subject, we select the five most frequently used skills in the training dataset to filter out skills that do not have enough training examples. The 7 subjects and the 5 selected skills for each are the following.

- **Algebra:** (1) Algebraic manipulation; (2) calculation and conversion; (3) arithmetic; (4) solving equations; (5) algebraic expression.
- **Counting and Probability:** (1) Number theory and arithmetic operations; (2) combinatorial mathematics; (3) counting principles; (4) probability concepts and calculations; (5) understanding and applying combinatorics concepts.
- **Geometry:** (1) Algebraic; (2) triangle geometry; (3) 3D geometry and volume calculation; (4) understanding circle properties and algebraic manipulation; (5) coordinate geometry and transformation.
- **Intermediate Algebra:** (1) Polynomial; (2) understanding and application of functions; (3) simplification and basic operations; (4) al-

	Method	Performance	% $\mathcal{T}_1$ CoT	% $\mathcal{T}_2$ CoT	% Both CoT
Next Letter + Mult	StandardCoT-Merge	70.4	85.3	95.1	85.3
	ComposableCoT-Merge	95.4	100.0	100.0	[†] 100.0
	StandardCoT-MTL	3.6	0.0	100.0	0.0
	ComposableCoT-MTL	96.3	98.9	100.0	[†] 98.9
Concat + Mult	StandardCoT-Merge	77.0	90.3	98.7	90.0
	ComposableCoT-Merge	75.4	91.6	100.0	91.6
	StandardCoT-MTL	72.1	99.7	32.1	32.1
	ComposableCoT-MTL	74.3	100.0	83.1	[†] 81.3
Concat + Next Letter	StandardCoT-Merge	54.8	100.0	99.4	99.4
	ComposableCoT-Merge	19.2	44.6	60.5	17.7
	StandardCoT-MTL	60.9	100.0	66.7	66.7
	ComposableCoT-MTL	63.3	100.0	85.9	[†] 85.0
Skill-Mix Literary + Rhetorical	StandardCoT-Merge	29.8	60.0	59.2	35.9
	ComposableCoT-Merge	39.6	64.1	66.9	[†] 43.3
	StandardCoT-MTL	42.0	65.3	58.0	37.6
	ComposableCoT-MTL	49.0	64.5	65.7	[†] 42.0

Table 16: Intrinsic evaluation of the generated CoTs from atomic CoT models evaluated on the compositional task in the zero-shot setting. “% $\mathcal{T}_1$ CoT” denotes the percentage of generated responses that use the CoT format of the first atomic task of the composition, and likewise for the second. [†] denotes that the ComposableCoT method has a significantly higher “% Both CoT” than the StandardCoT counterpart at the 0.01 level using a paired bootstrap test. Combined Composable CoT models generate responses including both atomic CoT patterns more frequently than combined atomic CoT models.

gebraic manipulation and equations; (5) complex number manipulation and operations.

- **Number Theory:** (1) Basic arithmetic; (2) understanding of fractions; (3) number manipulation; (4) number theory; (5) modular arithmetic.
- **Prealgebra:** (1) Counting and number theory; (2) solving linear equations; (3) basic arithmetic operations; (4) multiplication and division; (5) fractions and decimals.
- **Precalculus:** (1) Trigonometric calculations; (2) geometric relations; (3) algebra and equations; (4) vector operations; (5) geometry and space calculation.

An example of the atomic CoT data can be found in Example M.1.

M.1 Atomic Task Example: STAT-Algebra Example

[Subject]
Algebra
[Instruction]
Problem: Find $h(x)$, with terms in order of decreasing degree, if $3x^4 + 2x - 1 + h(x) = 5x^2 - 6x - 1$.
Answer:

[Chain-of-Thought + Answer String]
This equation is solved by
 $h(x) = (5x^2 - 6x - 1) - (3x^4 + 2x - 1) =$
 $\boxed{-3x^4 + 5x^2 - 8x}$
[Skill]
Solving Equation

Compositional tasks We consider each test example in the MATH dataset as a compositional example of the math reasoning skills discussed above. We follow He et al. (2025) to annotate the required skills for each MATH test example using GPT-4o-2024-11-20. An example of the compositional data can be found in Example M.2.

M.2 Compositional Task Example: STAT-Algebra Example

[Subject]
Algebra
[Instruction]
Problem: What is the smallest value of x such that $|5x - 1| = |3x + 2|$? Express your answer as a common fraction.
Answer:

Subject	# Train Examples
Algebra	250
Counting and Probability	300
Geometry	100
Intermediate Algebra	400
Number Theory	200
Prealgebra	100
Precalculus	200

Table 17: Training data distribution of the math skill composition setting.

[Chain-of-Thought + Answer String]
 There are two cases, when $5x - 1 = 3x + 2$ and when $5x - 1 = -(3x + 2)$. The two equations yield $x = \frac{3}{2}$ and $x = -\frac{1}{8}$, respectively, of which $x = \boxed{-\frac{1}{8}}$ is the smaller solution.

[Skill]
 Algebraic manipulation, Solving equations, Arithmetic

Difference from the Controlled Settings We note that the mathematical skill composition setting above is not a controlled compositional setting, as the string manipulation, arithmetic, and natural language skill composition settings in Section 4 are, for the following reasons.

(1) Atomic skills are not clearly separated from each other. As seen in the list of atomic skills above, many atomic skills overlap with each other within the same subject (e.g., “algebraic manipulation” and “algebraic expression” in the algebra subject).

(2) Compositions might be seen during pre-training. The task performance cannot be completely attributed to the composition of learned atomic skills, as the pre-trained capabilities of math reasoning might have also played a role.

M.2 Training

Standard CoT Data We use the *STAT-Syn* training dataset from He et al. (2025) and select the training examples that demonstrate the skills in the list above. We note that the original *STAT-Syn* dataset includes training examples of duplicate CoTs, and training on such examples makes models memorize specific CoT data verbatim. Thus, we pre-process the *STAT-Syn* data by de-duplication. The filtered dataset includes 100 – 500 training examples for each subject, which is consistent with our low-data

setting in Section 4. The distribution can be found in Table 17. We use the filtered dataset as the standard CoT dataset.

Composable CoT Data To augment the standard CoT dataset, we use the same data construction implementation used for two-way and three-way compositions of the string manipulation, arithmetic, and natural language skill tasks; see Section 4 and Appendix K.

Training Models We fine-tune an atomic CoT model for *each subject*, using the same training configurations described in Appendix H. We use multi-task learning to combine ComposableCoT models and StandardCoT models.

M.3 Inference

At inference time, we use the same inference scheme and configuration for the three-way compositions of the string manipulation, arithmetic, and natural language skill tasks, as described in Appendix K. Recall that at inference time, we assume knowledge of how many skills are required (i.e., the number of compositions) for each test example in the MATH test set, which ranges from 1 to 6-way compositions. We only evaluate each atomic CoT model on the MATH test set of the corresponding subject that it has been trained on; we do not perform cross-subject evaluation, as the atomic skills are subject-specific.

M.4 Evaluation

Evaluation Metric We use the lm-eval-harness library (Gao et al., 2023) to run all evaluations, and we evaluate all models on the test split of MATH following He et al. (2025). We use the answer parser from Hendrycks et al. (2021); Gao et al. (2023); Habib et al. (2023) to extract answers from the model-generated CoTs, and report exact match accuracy as the evaluation metric.

Performance by Subject Recall that we fine-tune a single model for each subject in MATH, since the atomic skills are specific to each subject. Therefore, we evaluate each subject-specific fine-tuned model on the test set of *only the corresponding subject*. We report the weighted average across subjects as the overall accuracy, because different subjects in the MATH test set have different numbers of test examples.

Performance Breakdown To evaluate the performance breakdown by the number of required skills,

as reported in Table 4, we follow He et al. (2025) to annotate the required skills for each MATH test example by GPT-4o-2024-11-20, and investigate the performance breakdown across test examples that: (1) Use at least 2 of the learned atomic skills; (2) Require 2–5 skills. We note that 6% of the test examples are not included in the performance breakdown rows, because they either use atomic skills that are not learned, or use 1 or more than 5 skills.

N Error Analyses

In addition to not being able to perform the individual atomic task correctly, we show three types of common errors made by ComposableCoT variants in the zero-shot compositional evaluation setting.

1. Example N.1 shows an example where the generated CoT is only able to replicate CoT of one atomic CoT and repeat the same CoT in the prefix and suffix.
2. Example N.2 shows an example where the combined model fails to continue generation after generating the prefix CoT. This is a common error for Composable models combined with model merging.
3. Example N.3 shows an example where the combined model uses the wrong atomic CoT in the prefix that should have been used in the suffix.

N.1 Error Case: Replicating One Atomic CoT Pattern

[Instruction]
Take the first letter of each word in the sequence, concatenate them in lower case, and find the next letter in alphabet following the last letter in the sequence of the concatenated letters: Jessica Berk Hamilton Stephanie answer:
[Answer String]
t
[Model Output]
<prefix> The last letter is e, and the letter following it in alphabet is f. So the answer is f.</prefix> <suffix> The last letter is e, and the letter following it in alphabet is f. So the answer is f.</suffix>

[Predicted Answer]
f

N.2 Error Case: Failing to Continue Generation after Prefix CoT

[Instruction]
Take the last letter of each word in the sequence, concatenate them in lower case, then find the ASCII value of the last letter in the sequence of the concatenated letters, and multiply the ASCII value by 6: Ayat Amaris Heavenly Mariela answer:
[Answer String]
582
[Model Output]
<prefix> The last letter of the 1st word is t. The last letter of the 2nd word is s. The last letter of the 3rd word is y. The last letter of the 4th word is a. So the answer is tasy, and the ASCII value of the last letter in the sequence of the concatenated letters is 121, so the answer is 726.</prefix>
<suffix>
[Predicted Answer]
726

N.3 Error Case: Use the Wrong Order of Atomic CoTs

[Instruction]
Take the second letter of each word in the sequence, concatenate them in lower case, and find the next letter in alphabet following the last letter in the sequence of the concatenated letters: D’shaun Takia Hiroki Arjun answer:
[Answer String]
s
[Model Output]
<prefix> the last letter is n, and the letter following it in alphabet is o. so the answer is o.</prefix> <suffix> the second letter of d is d, the second letter of t is a, the second letter of h is i, and the second letter of a is r. so the answer is dair.</suffix>
[Predicted Answer]
dair

O Computation resources

We conducted all our experiments primarily on a server with NVIDIA RTX A6000 48GB GPU, and a server with GPU of NVIDIA GH200 120GB. Each individual experiment can be run on a single A6000 of 48 GB within 1 hour.

P Licensing

We use the following publicly available datasets from prior works with open licenses.

Skill-Mix Yu et al. (2024) use the Apache-2.0 license and data is available at: <https://huggingface.co/spaces/dingliyu/skillmix>.

Letter concatenation The dataset uses the Apache-2.0 license and the data is available at: <https://huggingface.co/datasets/ChilleD/LastLetterConcat>

STAT-Syn He et al. (2025) use the Apache-2.0 license and data is available at: <https://github.com/princeton-qli/STAT>.