Compositional Generalization with Outcome-Level Optimization

Xiyan Fu[1] **and Wei Liu**[2*]
[1] Nanyang Technological University [2] Heidelberg Institute for Theoretical Studies
xiyan.fu@ntu.edu.sg | wei.liu.llm@gmail.com

Abstract

Compositional generalization refers to correctly interpret novel combinations of known primitives, which remains a major challenge. Existing approaches often rely on supervised fine-tuning, which encourages models to imitate target outputs. This token-level training paradigm fails to capture the global compositional structure required for generalizing to unseen combinations. In this work, we investigate whether compositional generalization can instead be improved through outcome-level reinforcement learning. We adopt Group Relative Policy Optimization to optimize models based on feedback on their final outputs. Within this framework, we explore both a simple binary outcome reward and a composite reward that provides additional composition feedback. Experiments on multiple compositional benchmarks show that reinforcement learning improves compositional generalization compared to supervised fine-tuning. Further analysis reveals that supervised models tend to overfit frequent training compositions, whereas reinforcement learning improves compositional generalization by reshaping the output distribution, particularly for more complex composition types.

1 Introduction

Compositional generalization (Fodor and Pylyshyn, 1988; Lake et al., 2017; Hupkes et al., 2020), a hallmark of human cognition, refers to the ability to understand and produce novel combinations by systematically recombining known components. As illustrated in Figure 1, humans who understand 'turn left and run → LTURN RUN' and 'walk and jump twice → WALK JUMP JUMP' can infer a new combination 'jump twice and turn left → JUMP JUMP LTURN' by recombining the known components. This ability enables extrapolation beyond the empirical data distribution and is therefore widely regarded as essential for models to generalize to unseen and increasingly complex reasoning tasks.

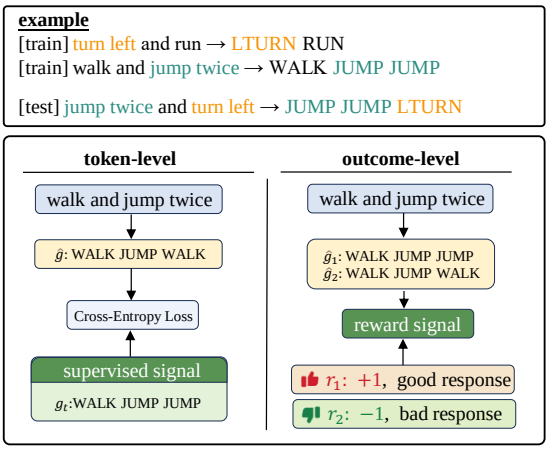


Figure 1: Illustration of compositional generalization and training paradigms. *Top*: Example of compositional generalization where the model must correctly compose previously seen primitives (e.g., jump twice and turn left) to produce the correct action sequence for a novel instruction. *Bottom*: Comparison of training signals. Token-level optimization relies on supervised targets and cross-entropy loss, whereas outcome-level optimization samples multiple responses and optimizes them using reward signals.

Given the importance of compositional generalization, extensive research has explored how to equip neural models with this capability. Early works (Lake and Baroni, 2018; Kim and Linzen, 2020) mainly relied on supervised fine-tuning (SFT) with teacher forcing (Lamb et al., 2016), where models are trained to imitate target outputs. However, such methods often struggle to generalize to novel compositions despite strong in-distribution performance. Subsequent studies have attempted to address this limitation through data augmentation (Andreas, 2020; Li et al., 2023; Yao and Koller, 2024), which exposes models to diverse compositional patterns, or architecture designs (Guo et al., 2020) that introduce compositional inductive biases. While these approaches improve performance on compositional benchmarks, they still rely on

* Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 31845–31857
October 24-29, 2026 ©2026 Association for Computational Linguistics

token-level supervision that optimizes local token predictions. However, compositional correctness is inherently a global property of the output sequence, i.e., an output may appear locally plausible while still violating the intended composition rule. This mismatch raises a natural question: *can compositional generalization be learned through outcome-level optimization* instead?

To explore this question, we adopt a reinforcement learning (RL) framework, in which models are optimized based on feedback on their final outputs rather than token-level supervision. Specifically, we employ Group Relative Policy Optimization (GRPO) (Shao et al., 2024) as the training framework. Figure 1 compares the training signals used in SFT and GRPO. Within this framework, we design two types of reward signals: (i) a coarse-grained binary reward that provides feedback based on the exact match between the generated output and the ground truth, and (ii) a fine-grained composite reward that decomposes the task and provides additional feedback for primitive prediction and compositional structure. Across multiple compositional benchmarks, we find that RL training improves compositional generalization performance. Interestingly, we observe that two rewards achieve comparable results, suggesting that compositional generalization may emerge from minimal outcome-level feedback.

To better understand differences between token-level supervision and outcome-level optimization in composition, we analyze the behaviors of models trained with SFT and RL. We investigate performance across compositional types, finding larger improvements from RL on productivity tests (i.e., length splits), particularly for compositions closer to the training length. Furthermore, our analysis reveals that SFT-trained models tend to overfit compositional patterns observed during training, often reproducing familiar structures when they make incorrect predictions for novel compositions. In contrast, RL-trained models reshape the output probability distribution, reducing the bias toward frequent training compositions and improving the ranking of candidates. This behavior becomes evident under multi-sample evaluation (e.g., pass@k), where RL mainly improves top-1 accuracy while achieving comparable performance at larger k.

Our main contributions are as following:

- We propose reinforcement learning paradigm for compositional generalization, where models are optimized using feedback on the final output rather than token-level supervision.
- We design two types of reward signals, binary and composite, to guide RL training and improve compositional generalization across multiple benchmarks.
- Through behavioral analyses, we demonstrate that SFT tends to overfit training patterns. In contrast, RL reshapes the output distribution and leads to improvements in compositional generalization ability, especially on complex composition types.

2 Related Work

Compositional Generalization Compositional generalization refers to the ability of a model to systematically recombine known primitives or rules to understand or generate novel compositions (Fodor and Pylyshyn, 1988; Lake et al., 2017; Hupkes et al., 2020). It has been extensively studied across diverse settings, such as in semantic parsing (Lake and Baroni, 2018; Kim and Linzen, 2020; Qiu et al., 2022b), machine translation (Li et al., 2021; Dankers et al., 2022), deductive reasoning (Saparov et al., 2023), natural language inference (Yanaka et al., 2020; Fu and Frank, 2023), etc. They found that even state-of-the-art LLMs are still not able to perform compositional generalization.

To address this limitation, a number of approaches have been proposed. Qiu et al. (2022a); Levy et al. (2023); Yao and Koller (2024) introduce data augmentation strategies that inject compositional inductive bias into neural sequence models by generating additional training examples or recombining components across existing examples. Other works (Zheng and Lapata, 2021; Herzig and Berant, 2021) design specialized model architectures to better capture compositional structures, such as modular architectures or parallel decoding mechanisms. More recent studies explore alternative training paradigms, including meta-learning (Conklin et al., 2021; Lake and Baroni, 2023) and continual learning (Fu and Frank, 2024), to improve models' adaptability to more realistic scenarios. However, these approaches still rely on token-level supervision, optimizing models to predict individual tokens rather than ensuring the correctness of the entire compositional output sequence. In contrast, we investigate whether compositional generalization can be improved through reinforcement

learning, which provides outcome-level feedback and enables global optimization over complete sequences.

Reinforcement Learning for Generalization Reinforcement learning is widely used in the post-training of LLMs to better align model outputs with human preferences (Ouyang et al., 2022; OpenAI, 2023). Beyond alignment, recent studies suggest that RL can also improve generalization compared to imitative learning (Bi et al., 2024; Lambert et al., 2025). In particular, RL tuning enhances cross-lingual generalization, especially when trained on non-English data (Huang et al., 2026), as well as cross-category generalization, where training on one domain (e.g., geometry) transfers to others (e.g., algebra) (Wang et al., 2025). It also improves cross-domain generalization, e.g., Wei et al. (2025) introduce Llama3-SWE-RL, which achieves strong performance on software engineering tasks despite being trained solely on software evolution data. Similar trends are observed in multimodal settings, where Chu et al. (2026) report improved generalization across both unimodal and multimodal reasoning benchmarks. More recently, several studies have explored modifications to the RL training paradigm to further enhance generalization. For example, Tang et al. (2025); Zhang et al. (2026) combine on-policy and off-policy learning to improve the generalizability of reasoning language models.

In contrast to these works, we focus on compositional generalization, investigating how RL affects the ability of models to handle novel combinations of known primitives. In addition, we conduct a detailed comparison between RL and supervised fine-tuning (SFT), showing that RL reshapes distributions and avoids copying frequent patterns of training compositions.

3 Methods

3.1 Problem Definition

Compositional generalization tests are designed to evaluate whether a model can generalize to unseen compositions whose constituting primitives and composing rules have been observed in training. For example, we can evaluate a model's compositional generalization ability by testing it on an unseen compositional sample *jump twice and turn left* $\rightarrow$ *JUMP JUMP LTURN* after training with *jump twice after run* $\rightarrow$ *RUN JUMP JUMP* and *run and turn left* $\rightarrow$ *RUN LTURN*. Here, *jump* $\rightarrow$ *JUMP* and *turn left* $\rightarrow$ *LTURN* are primitives, v_x *and* v_y $\rightarrow$ *X Y* is the implicit composition rule. We denote the set of primitives as $\mathcal{P}$ and the set of composition rules as $\mathcal{R}$. A compositional instance is generated by applying a composition rule $r \in \mathcal{R}$ to a tuple of primitives $(p_1, \ldots, p_k)$, where $p_i \in \mathcal{P}$. Each compositional type can therefore be represented as a pair $(\mathbf{p}, r)$, where $\mathbf{p}$ denotes the primitives involved. Correspondingly, $\mathcal{C}$ denotes the set of all possible compositional types:

$$\mathcal{C} = \{(\mathbf{p}, r) \mid \mathbf{p} \subseteq \mathcal{P},\ r \in \mathcal{R}\}. \tag{1}$$

For compositional generalization tests, the sets of compositional types used for training and testing are always ensured disjoint, $\mathcal{C}_{\text{train}} \cap \mathcal{C}_{\text{test}} = \emptyset$. At the same time, all primitives and composition rules appearing in test instances are guaranteed to have been observed during training: $\mathcal{P}_{\text{test}} \subseteq \mathcal{P}_{\text{train}}$, $\mathcal{R}_{\text{test}} \subseteq \mathcal{R}_{\text{train}}$. This setup ensures that test instances require genuine recombination of known primitives under familiar composition rules, rather than extrapolation to entirely unseen components.

3.2 Compositional Group Relative Policy Optimization

To investigate whether compositional generalization can be learned through outcome-level optimization, we adopt a reinforcement learning (RL) training framework. This paradigm allows LLMs to autonomously explore and discover latent compositional structures through reward-based feedback. Specifically, we employ Group Relative Policy Optimization (GRPO) (Shao et al., 2024), a computationally efficient RL algorithm that has demonstrated strong empirical performance. The overall training procedure is illustrated in Figure 2.

3.2.1 Policy and Rollout Generation

We parameterize the language model as a stochastic policy $\pi_\theta(y \mid x)$, which generates an output sequence y conditioned on an input x. For each training input, we sample a group of K candidate outputs, forming a rollout group $\mathcal{G}$ associated with the same input.

$$\mathcal{G} = \{y^1, \ldots, y^K\} \sim \pi_\theta(\cdot \mid x), \tag{2}$$

Each output y^i in the group is a complete compositional prediction generated by the model.

3.2.2 Reward Assignment

For each sampled output y^i, we assign a scalar reward r^i based on the correctness of the generated

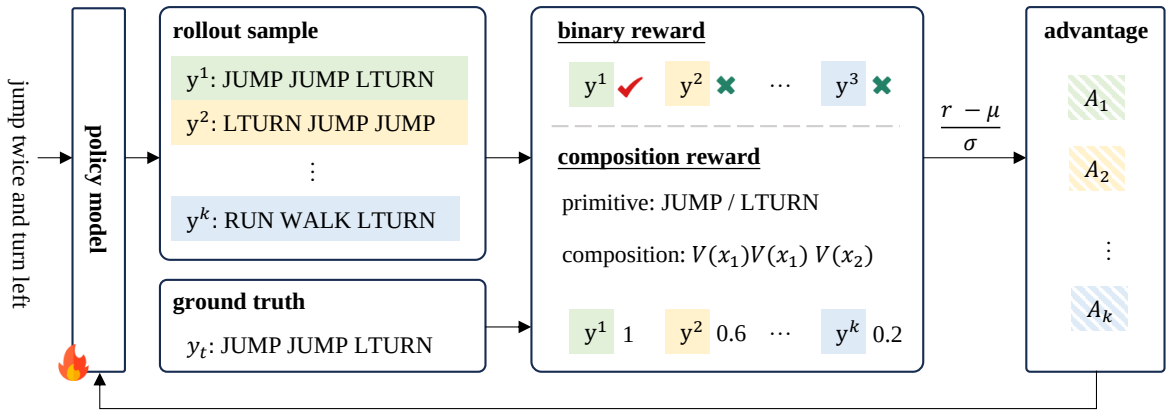


Figure 2: Overview of Compositional Group Relative Policy Optimization. The LLM samples a group of candidate label sequences $(y^1, y^2, \ldots, y^K)$ based on inputs. Each candidate is then evaluated using two complementary reward signals: (i) a *binary reward* that measures exact match with the gold sequence, and (ii) a *compositional reward* that assesses primitive correctness and structural composition patterns. Rewards are normalized within the sampled group to compute relative advantages, which are then used to update the policy via group-relative optimization.

prediction. We consider two types of reward functions: a *binary reward* that provides coarse-grained supervision, and a *composite reward* that offers more fine-grained feedback.

Binary Reward. The binary reward assigns a score of either 0 or 1 based on exact matching between the generated compositional prediction and the ground-truth output. While simple and deterministic, this binary reward treats all incorrect trajectories uniformly, failing to capture partial correctness or relative quality among different erroneous predictions.

Composite Reward. To provide more fine-grained feedback for incorrect predictions, we introduce a composite reward that evaluates both *primitive correctness* and *compositional structure*. Given a predicted action sequence y^{pred} and a ground-truth sequence y^{gold}, we define the two reward components as follows:

- **Primitive.** We extract the sets of primitives appearing in the predicted and gold sequences, denoted as P^{pred} and P^{gold}, respectively. The primitive reward measures the coverage of gold primitives by the prediction, denoted as:

$$R_{\text{prim}}(y^{\text{pred}}, y^{\text{gold}}) = \frac{\left|P^{\text{pred}} \cap P^{\text{gold}}\right|}{\left|P^{\text{gold}}\right|}. \quad (3)$$

  This reward encourages the model to correctly identify and generate the required primitives, while ignoring invariant to their ordering and repetition. For example, in the compositional instruction example, *jump twice and turn left* → *JUMP JUMP LTURN*, the primitives JUMP and LTURN are extracted.

- **Composition.** We abstract away concrete primitive identities and focus solely on their compositional patterns. Each primitive is mapped to a variable, yielding a normalized and ordered sequence that represents the compositional skeleton. We then compute the compositional reward by measuring position-wise matches between the predicted and gold skeletons. Let C^{pred} and C^{gold} denote the skeletons derived from the predicted and gold sequences, respectively. The compositional reward is defined as:

$$R_{\text{comp}}(y^{\text{pred}}, y^{\text{gold}}) = \frac{1}{|C^{\text{gold}}|} \sum_{i=1}^{|C^{\text{gold}}|} \mathbf{1}\left[C_i^{\text{pred}} = C_i^{\text{gold}}\right] \quad (4)$$

  This reward encourages the model to focus on compositional structures, while ignoring primitives. For instance, the instruction *jump twice and turn left* → *JUMP JUMP LTURN* yields the compositional skeleton $V(x_1)\ V(x_1)\ V(x_2)$, obtained by mapping each distinct primitive to a unique placeholder variable.

The final composite reward [1] is obtained as the weighted sum of the primitive and composition components. We denoted it as $R = \lambda_1 R_{\text{prim}} + \lambda_2 R_{\text{comp}}$, where λ_1 and λ_2 are hyper-parameters.

[1] The specific definitions of primitives and compositions for each dataset are provided in Table 4 (Appendix A.2).

3.3 Group Relative Policy Optimization

To optimize the policy, we employ Group Relative Policy Optimization (GRPO), where the advantage is computed by comparing the relative performance of trajectories within the same sampled group $\mathcal{G}$. Specifically, we compute a normalized advantage for each sampled output by subtracting the group mean reward:

$$\hat{A}_i = \frac{r_i - \mu_r}{\sigma_r}. \tag{5}$$

where μ_r and σ_r denote the mean and standard deviation of rewards within the rollout group. This group-relative advantage emphasizes which generations perform better or worse compared to other candidates under the same input, mitigating reward scale sensitivity and stabilizing training.

Given the computed advantages, the policy is updated by maximizing the expected advantage-weighted log-likelihood of generated outputs while constraining excessive deviation from a reference policy. The GRPO objective is defined as:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}\left[\frac{1}{K}\sum_{i=1}^{K}\frac{1}{|y^i|}\sum_t \mathcal{L}_{\text{clip}}\big(r_{i,t}(\theta), \hat{A}_{i,t}\big)\right] - \beta D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}) \tag{6}$$

The clipping operator limits excessively large policy updates with threshold ϵ, denoted as:

$$\mathcal{L}_{\text{clip}}(r, \hat{A}) = \min(r\hat{A}, \text{clip}(r, 1-\epsilon, 1+\epsilon)\hat{A}) \tag{7}$$

Here, K denotes the number of sampled outputs in each group, and $|y^i|$ represents the length of i-th generated output y^i. The term $r_{i,t}(\theta)$ denotes the probability ratio between the current policy π_θ and the behavior policy used to generate the samples, while $\hat{A}_{i,t}$ represents the token-level advantage which is uniformly broadcast from $\hat{A}_i$ defined in Eq. 5. The KL divergence with strength β regularizes the policy, preventing excessive deviation from the supervised distribution.

4 Setup

4.1 Datasets

We evaluate on four standard compositional generalization benchmarks: SCAN, COGS, GeoQuery, and CFQ. They capture complementary aspects of compositionality and form a diverse testbed for evaluation. Data statistics and examples of primitive and compositional structures for reward design are provided in Appendix A.1 and A.2.

SCAN (Lake and Baroni, 2018) is a synthetic dataset designed to test compositional generalization in grounded language understanding. Each example pairs a simple navigation command with a sequence of atomic actions representing its execution, e.g., *jump twice and turn left* → *JUMP JUMP LTURN*. For composite reward, we define primitives at the action level (e.g., JUMP, LTURN), while compositional structures are abstracted as higher-level templates over primitives (e.g., $V(x_1)$ $V(x_1)$ $V(x_2)$). SCAN provides several splits targeting different compositional generalization settings. We select the *turnleft* and *length* splits, as they are more challenging (Qiu et al., 2022a).

COGS (Kim and Linzen, 2020) is a benchmark designed to evaluate compositional generalization in semantic interpretation. The task is to map natural language sentences to structured semantic representations, e.g., *A hedgehog ate the cake* → $\times$ *cake(x_4) ; hedgehog(x_1) AND eat.agent(x_2, x_1) AND eat.theme(x_2, x_4)*. For composite reward, we define predicates and entities as primitives (e.g., hedgehog, cake, eat.agent, eat.theme) and abstract their underlying argument structures into compositional templates (e.g., $N(x_4) \wedge N(x_1) \wedge V.agent(x_2, x_1) \wedge V.theme(x_2, x_4)$).

GeoQuery (Herzig and Berant, 2021) is a semantic parsing benchmark that maps natural language questions to logical forms representing queries over a geographic database. Each example consists of a question paired with a corresponding structured query, e.g., *What states border Texas?* → *answer state next_to stateid texas*. For compositional reward design, we decompose logical forms into atomic primitives, including entities and relations (e.g., stateid(texas), next_to), and abstract their compositional templates (e.g., $N(x_1) \wedge R(x_1, x_2)$). We use the linearized FunQL representation provided in the dataset, where logical forms are expressed as sequences of function tokens.

CFQ (Keysers et al., 2020) is a semantic parsing benchmark for evaluating compositional generalization in mapping natural language questions to SPARQL queries. Each instance consists of a question paired with its corresponding logical form, e.g., *Who directed Elysium?* → *SELECT DISTINCT ?x0 WHERE {?x0 a ns:people.person. ?x0 ns:film.director.film m.0gwm_wy .}*. For composite reward, we further decompose SPARQL queries into atomic primitives as entities and relations

	SCAN		COGS	GeoQuery		CFQ			
	length	turn left		template	length	MCD1	MCD2	MCD3	Avg
Llama									
SFT	11.88±0.22	69.03±0.32	81.12±0.44	50.36±0.55	43.21±0.48	91.04±0.10	74.85±0.17	77.42±0.19	81.10
GRPO-Binary	18.49±0.32	71.52±0.23	83.90±0.56	52.48±0.42	45.92±0.40	91.35±0.15	76.44±0.23	81.46±0.18	83.08
GRPO-Compos	23.44±0.15	70.74±0.18	82.88±0.40	52.07±0.59	46.17±0.31	92.10±0.29	76.16±0.14	83.44±0.16	83.90
Qwen									
SFT	18.41±0.31	78.06±0.29	82.09±0.36	73.65±0.36	28.21±0.59	88.52±0.13	73.98±0.13	73.67±0.53	78.72
GRPO-Binary	22.68±0.22	79.13±0.25	83.23±0.37	75.27±0.35	32.14±0.54	89.23±0.29	75.73±0.33	76.54±0.27	80.50
GRPO-Compos	21.81±0.14	78.76±0.23	83.15±0.28	77.90±0.51	33.76±0.45	89.16±0.20	77.33±0.18	75.86±0.19	80.78

Table 1: Results of SFT and GRPO on different compositional generalization benchmarks across various settings. We report exact matching accuracy for all evaluations. All results are averaged over three runs and reported as mean accuracy with standard deviation.

(e.g., ns:people.person, ns:film.director.film), and abstract the composition structures (e.g., $N(x_1) \wedge R(x_1, x_2)$). We adopt the three Maximum Compound Divergence (MCD) splits provided in the original dataset, which are specifically designed to induce strong compositional distribution shifts.

4.2 Implementation Details

We conduct our experiments on Qwen-2.5-7B-Instruct (Qwen, 2024) and Llama-3.1-8B-Instruct (Grattafiori et al., 2024), given their superior performance. During training, we observe that reinforcement learning from scratch suffers from unstable optimization due to the large and sparse exploration space. Therefore, we adopt a supervised warm-up stage[2] via SFT to initialize the policy following previous works (Shao et al., 2024), which significantly stabilizes subsequent GRPO training.[3] After the warm-up stage, we further optimize the policy using Compositional Group Relative Policy Optimization. For each input, we sample 8 trajectories with a temperature of 0.6. The update batch size is 8, and the learning rate is set to 1e-6. The hyperparameters λ_1 and λ_2 for the composite reward are set to 0.1 and 0.2, respectively.[4]

5 Results

5.1 Main Results

Table 1 reports the compositional generalization results across four benchmarks using two base models. Compared with the SFT baseline, both GRPO variants, GRPO-Binary and GRPO-Composite, consistently achieve better performance across all benchmarks and model bases. The results suggest that *outcome-level optimization (i.e., RL-based strategies) is more effective than token-level imitation (i.e., SFT-based strategies) for learning compositional behaviors*.

We further analyze the results across different types of compositional generalization.[5] We observe that the improvements brought by GRPO are generally larger on productivity settings that require generating longer compositions of known primitives. For example, on the SCAN length split, GRPO-Binary improves performance by 6.61% and 4.27% for the two base models, respectively. In contrast, the gains are smaller on systematicity evaluations, where generalization requires recombining known primitives into novel structures. For instance, on the MCD3 split of CFQ, which exhibits the largest generalization gap, the improvements are 4.04% and 2.87%. Similar trends are observed across GRPO-Composite strategies. Notably, the absolute performance on productivity tasks (e.g., length splits) is substantially lower than that on systematicity evaluations, indicating that productivity compositional generalization is considerably more challenging. These results suggest that outcome-level optimization exhibits different benefits across compositional generalization types and is particularly effective for more challenging settings such as productivity tasks.

In addition, we compare different GRPO reward strategies and observe that GRPO-Binary and GRPO-Compos achieve comparable performance across various settings. This suggests that compositional generalization tests are relatively insensitive to the specific reward formulation. That is, a simple binary outcome signal is already sufficient to guide

[2]The details of warmup are provided in the Appendix B

[3]For GRPO training, we use the code `https://github.com/verl-project/verl`.

[4]Details of hyperparameters are provided in Appendix C.

[5]Different compositional generalization types are discussed in Hupkes et al. (2020)

	CFQ-MCD3			SCAN-Length		
	prim	com	em	prim	com	em
R_{P+C}	98.66	90.46	83.44	97.96	92.41	23.44
R_P	98.58	88.43	81.60	97.81	89.72	20.97
R_C	98.32	90.19	82.73	97.19	92.15	22.13

Table 2: Ablation study of GRPO-Compos training paradigm based on Llama-3.1-Instruct. We decompose the composite reward design R_{P+C} by only offering primitive reward R_P and composite reward R_C. *prim* and *com* represent primitive and composition accuracy, respectively. *em* denotes the exact matching accuracy.

effective policy optimization.

5.2 Ablation Study

In this section, we conduct an ablation study to investigate the effect of compositional reward design in the GRPO-Compos training paradigm on two challenging compositional generalization settings, i.e., MCD3 of CFQ and length of SCAN. Specifically, we decompose the composite reward into its individual components and train GRPO with only one reward at a time: i) the primitive reward, which provides positive feedback for correctly predicting primitives, and ii) the composition reward, which rewards correct compositional structures. The full GRPO-Compos model combines both signals. Table 2 reports the ablation results.

We observe that the full composite reward consistently achieves higher exact-match accuracy than using either reward alone. This indicates that the two reward signals provide complementary information for learning compositional generalization. To better understand the effect of different reward signals, we further decompose the evaluation metrics into primitive accuracy and composition accuracy. Their definitions are provided in Eq. 3 and Eq. 4. As shown in Table 2, primitive reward primarily improves primitive accuracy, while composition reward contributes more directly to composition accuracy. Interestingly, primitive accuracy remains comparable between primitive reward only (R_P) and composite reward only (R_C), suggesting that composition reward can also indirectly enhance primitive prediction. Overall, these results highlight a non-trivial interaction between reward design and the learning dynamics in RL.

6 Analysis

In this section, we further compare results from the SFT and GRPO training paradigms to better understand how they influence compositional generalization.

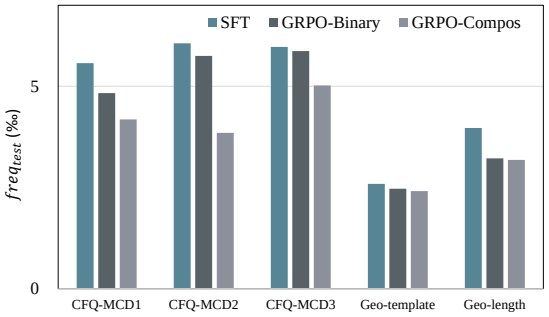


Figure 3: Average training trigram frequency of incorrect predictions from SFT and GRPO. Trigram frequencies are computed with respect to the training data, excluding trigrams appearing in ground-truth outputs.

6.1 Copying Behavior

Prior works (Rice et al., 2020) suggest that poor generalization may stem from models overfitting to the prior output distribution of the training data. In compositional generalization tasks, this manifests as models relying on frequently occurring output fragments rather than composing primitives according to the input. In this section, we examine whether token-level optimization (i.e., SFT) encourages models to follow such frequent structural patterns, and whether outcome-level optimization (i.e., GRPO) can alleviate this issue.

To quantify this tendency, we examine whether model predictions prefer output trigrams that are more frequent in the training data. We first compute trigram statistics from the output sequences in the training set and normalize their frequencies, denoted as $freq_{train}$. For incorrect samples, we extract all trigrams from the model predictions and look up their frequencies according to $freq_{train}$. We then define the average trigram frequency of a sequence as a measure of how strongly it aligns with common structural patterns in the training data. To avoid bias from valid output patterns, we exclude trigrams that also appear in the ground-truth outputs when computing the average trigram frequency. Finally, we compare this metric for incorrect samples produced by SFT and GRPO. We conduct this analysis on CFQ and GeoQuery, where trigrams capture meaningful structural patterns. The results are shown in Figure 3.

We observe that the SFT paradigm consistently produces predictions with higher average trigram frequency than GRPO. This suggests that SFT

	CFQ-MCD3			SCAN-Length		
	@1	@5	@10	@1	@5	@10
SFT	77.90	82.21	83.28	11.02	20.69	23.77
GRPO-B	81.57	82.66	83.42	18.57	21.10	23.81
GRPO-C	83.42	83.53	83.61	23.85	24.21	24.39

Table 3: Comparison of top-k accuracy (pass@k) for SFT and GRPO-based methods, i.e., GRPO-B(inary) and GRPO-C(omposite).

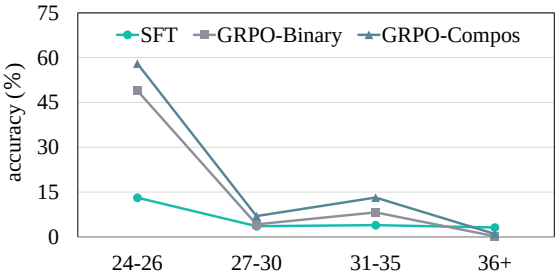


Figure 4: Performance on the SCAN-Length split across various target output lengths. Examples are grouped into bins by output length.

tends to rely more on frequent structural patterns from the training data, whereas GRPO exhibits weaker alignment with such patterns. Furthermore, we find that GRPO-Compos yields lower trigram frequencies than GRPO-Binary, particularly on the CFQ benchmark, even though it does not lead to higher accuracy (see Table 1). This result suggests that rewarding primitives and compositional structures helps mitigate the tendency to copy frequent patterns from the training data.

6.2 Top-k Performance

The trigram analysis suggests that SFT tends to rely more heavily on frequent patterns from the training data. Such reliance may affect the ranking of candidate outputs during decoding, causing inferior compositional generalization ability. To further examine this effect, we evaluate models using top-k accuracy. Specifically, for each test example, we set the sampling temperature to 0.6 and generate k candidate outputs. We then measure whether at least one sampled output matches the gold output, denoted as *pass@k*. Formally, $\text{pass@k} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\left[\exists j \leq k,\ \hat{y}_i^{(j)} = y_i\right]$, where y_i denotes the gold output and $\hat{y}_i^{(j)}$ is the j-th sampled prediction. Table 3 presents results.

We find that GRPO variants achieve higher top-1 accuracy (pass@1) than SFT on both benchmarks, while their top-10 performance (pass@10) remains comparable. This suggests that GRPO mainly improves the ranking of candidate outputs by assigning higher probability to correct structures, resulting in a sharper output distribution. In contrast, SFT tends to rank frequent training patterns higher, which may lead to incorrect predictions being prioritized during decoding. Consistently, the larger gap between pass@1 and pass@k of SFT indicates greater sensitivity to the choice of k, whereas GRPO remains comparatively stable across different decoding budgets.

6.3 Length Extrapolation

To further understand how different training paradigms behave under varying compositional complexity, we analyze model performance with respect to output length. In compositional generalization tasks, longer outputs typically correspond to more complex structural compositions and therefore present a more challenging compositionality. We group the test examples into several buckets according to the gold output length (24–26, 27–30, 31–35, and 36+) and report the corresponding pass@k performance for each bucket.

Figure 4 shows the accuracy of SFT and GRPO on SCAN-length as output length increases. All methods exhibit a general decline beyond lengths of 24–26, indicating that compositional generalization degrades with longer outputs and remains challenging even under outcome-level optimization. GRPO consistently outperforms SFT, particularly on shorter outputs (24–26 tokens), suggesting that SFT struggles to produce structurally correct outputs even at shorter lengths. In contrast, GRPO mitigates this issue through reward-based optimization, encouraging structurally valid predictions. Moreover, GRPO-Compos achieves slightly better performance, indicating that explicitly rewarding primitives and compositional structures further enhances the model's ability to capture compositional patterns.

6.4 Error Analysis

To further investigate why GRPO-based methods still underperform on compositional semantic parsing tasks, we conduct an error analysis. We sampled 200 incorrect predictions from CFQ-MCD3 for error analysis, with samples drawn proportionally to the error rates of SFT and GRPO. Since SFT and GRPO have error rates of 22.58% and 18.54%,

respectively, this roughly gives 110 SFT errors and 90 GRPO errors. They are classified into four error types: i) syntactic errors, where the output cannot be parsed; ii) semantic-unit errors, where the model predicts incorrect, missing, or extra entities, relations, or predicates; iii) structure errors, where the syntactically valid units are combined with incorrect nesting, argument binding, variable sharing, or relation structure; iv) others.

The analysis shows that GRPO mainly reduces syntactic and semantic-unit errors. Specifically, syntactic errors decrease from 24% to 17%, and semantic-unit errors decrease from 18% to 12% over the full test set. This suggests that outcome-level optimization mitigates SFT's tendency to prefer locally plausible but globally incorrect outputs under compositional shifts. Structure errors remain the dominant source of failures, accounting for 41% of the remaining incorrect cases in GRPO, and 43% in SFT. The pass@k analysis suggests that RL mainly improves ranking when better candidates are already sampled. For harder compositional-structure cases, it is hard to sample the potential correct answers, limiting the positive RL signal.

7 Conclusion

In this work, we study whether compositional generalization can be improved through outcome-level optimization rather than token-level supervised training. Experiments across multiple benchmarks show that GRPO consistently improves compositional generalization. Additionally, we find that a simple binary reward performs comparably to composite rewards. Further analysis suggests that RL reshapes the distribution and mitigates overfitting to training compositions, thereby improving compositional generalization.

8 Limitation

Reinforcement learning introduces additional computational cost compared to standard supervised fine-tuning, which may limit its practicality in large-scale training scenarios. In addition, our study only explores relatively simple reward formulations, including binary and composite rewards. More sophisticated reward designs or alternative outcome-level feedback signals may further affect the effectiveness of reinforcement learning for compositional generalization, which we leave for future work.

References

Jacob Andreas. 2020. Good-enough compositional data augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7556–7566, Online. Association for Computational Linguistics.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. 2024. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*.

Xiangxiang Chu, Hailang Huang, Xiao Zhang, Fei Wei, and Yong Wang. 2026. GPG: A simple and strong reinforcement learning baseline for model reasoning. In *The Fourteenth International Conference on Learning Representations*.

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 and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3322–3335, Online. Association for Computational Linguistics.

Verna Dankers, Elia Bruni, and Dieuwke Hupkes. 2022. The paradox of the compositionality of natural language: A neural machine translation case study. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4154–4175, Dublin, Ireland. Association for Computational Linguistics.

Jerry A Fodor and Zenon W Pylyshyn. 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71.

Xiyan Fu and Anette Frank. 2023. SETI: Systematicity evaluation of textual inference. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4101–4114, Toronto, Canada. Association for Computational Linguistics.

Xiyan Fu and Anette Frank. 2024. Exploring continual learning of compositional generalization in NLI. *Transactions of the Association for Computational Linguistics*, 12:912–932.

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

Yinuo Guo, Zeqi Lin, Jian-Guang Lou, and Dongmei Zhang. 2020. Hierarchical poset decoding for compositional generalization in language. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6913–6924. Accepted at NeurIPS 2020.

Jonathan Herzig and Jonathan Berant. 2021. Span-based semantic parsing for compositional generalization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 908–921, Online. Association for Computational Linguistics.

Shulin Huang, Yiran Ding, Junshu Pan, and Yue Zhang. 2026. Beyond english-centric training: How reinforcement learning improves cross-lingual reasoning in LLMs. In *The Fourteenth International Conference on Learning Representations*.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. 2020. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795.

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

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

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International conference on machine learning*, pages 2873–2882. PMLR.

Brenden M Lake and Marco Baroni. 2023. Human-like systematic generalization through a meta-learning neural network. *Nature*, pages 1–7.

Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. 2017. Building machines that learn and think like people. *Behavioral and brain sciences*, 40.

Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. *Advances in neural information processing systems*, 29.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, and 4 others. 2025. Tulu 3: Pushing frontiers in open language model post-training. In *Second Conference on Language Modeling*.

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)*, pages 1401–1422, Toronto, Canada. Association for Computational Linguistics.

Yafu Li, Yongjing Yin, Yulong Chen, and Yue Zhang. 2021. On compositional generalization of neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4767–4780, Online. Association for Computational Linguistics.

Zhaoyi Li, Ying Wei, and Defu Lian. 2023. Learning to substitute spans towards improving compositional generalization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2791–2811, Toronto, Canada. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Linlu Qiu, Peter Shaw, Panupong Pasupat, Pawel Nowak, Tal Linzen, Fei Sha, and Kristina Toutanova. 2022a. Improving compositional generalization with latent structure and data augmentation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4341–4362, Seattle, United States. Association for Computational Linguistics.

Linlu Qiu, Peter Shaw, Panupong Pasupat, Tianze Shi, Jonathan Herzig, Emily Pitler, Fei Sha, and Kristina Toutanova. 2022b. Evaluating the impact of model scale for compositional generalization in semantic parsing. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9157–9179, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Team Qwen. 2024. Qwen2.5: A party of foundation models.

Leslie Rice, Eric Wong, and Zico Kolter. 2020. Overfitting in adversarially robust deep learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8093–8104. PMLR.

Abulhair Saparov, Richard Yuanzhe Pang, Vishakh Padmakumar, Nitish Joshi, Mehran Kazemi, Najoung Kim, and He He. 2023. Testing the general deductive reasoning capacity of large language models using OOD examples. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y.K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models.

Yunhao Tang, Taco Cohen, David W Zhang, Michal Valko, and Rémi Munos. 2025. Rl-finetuning llms from on-and off-policy data with a single algorithm. *arXiv preprint arXiv:2503.19612*.

Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Liyuan Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, Weizhu Chen, Shuohang Wang, Simon Shaolei Du, and yelong shen. 2025. Reinforcement learning for reasoning in large language models with one training example. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, LINGMING ZHANG, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida Wang. 2025. SWE-RL: Advancing LLM reasoning via reinforcement learning on open software evolution. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Hitomi Yanaka, Koji Mineshima, Daisuke Bekki, and Kentaro Inui. 2020. Do neural models learn systematicity of monotonicity inference in natural language? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6105–6117, Online. Association for Computational Linguistics.

Yuekun Yao and Alexander Koller. 2024. Simple and effective data augmentation for compositional generalization. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 434–449, Mexico City, Mexico. Association for Computational Linguistics.

Wenhao Zhang, Yuexiang Xie, Yuchang Sun, Yanxi Chen, Guoyin Wang, Yaliang Li, Bolin Ding, and Jingren Zhou. 2026. On-policy RL meets off-policy experts: Harmonizing supervised fine-tuning and reinforcement learning via dynamic weighting. In *The Fourteenth International Conference on Learning Representations*.

Hao Zheng and Mirella Lapata. 2021. Compositional generalization via semantic tagging. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1022–1032, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Dataset	Setting	Example
SCAN	original	jump twice and turn left → JUMP JUMP LTURN
	primitive	JUMP LTURN
	composition	$V(x_1)\ V(x_1)\ V(x_2)$
COGS	original	A hedgehog ate the cake → cake(x_4) ; hedgehog(x_1) AND eat.agent(x_2, x_1) AND eat.theme(x_2, x_4)
	primitive	[*entity prim*] hedgehog, cake [*relation prim*] eat.agent, eat.theme
	composition	$N(x_4)\ \wedge\ N(x_1)\ \wedge\ V.agent(x_2, x_1)\ \wedge\ V.theme(x_2, x_4)$
GeoQuery	original	What states border Texas? → answer state next_to stateid texas
	primitive	[*entity prim*] texas [*relation prim*] next_to
	composition	$N(x_1)\ \wedge\ R(x_2, x_1)$
CFQ	original	Who directed Elysium? → SELECT DISTINCT ?x0 WHERE { ?x0 a ns:people.person. ?x0 ns:film.director.film m.0gwm_wy .}
	primitive	[*entity prim*] ns:people.person, m.0gwm_wy [*relation prim*] ns:film.director.film
	composition	$N(x_1)\ \wedge\ R(x_1, x_2)$

Table 4: Examples from SCAN, COGS, GeoQuery, and CFQ illustrating primitive–composition decomposition. For each dataset, we show the original input–output pair and its decomposition into primitives and abstract composition structures. This formulation separates lexical primitives from structural composition, enabling compositional-level reward. We abstract primitives into entities/actions and relations, and represent compositional structures using unified predicate templates $N(x)$, $R(x, y)$, and $V(x)$.

A Data

A.1 Data Statistic

We use four compositional benchmarks in this paper, including SCAN, COGS, GeoQuery, and CFQ. The data statistic is collected in Table 5.

Dataset	Setting	# train	# dev.	# test
SCAN	turnleft	21890	-	1208
	length	16990	-	3920
COGS	-	24155	3000	3000
GeoQuery	template	544	60	276
	length	540	60	280
CFQ	MCD1	95743	11968	11968
	MCD2	95743	11968	11968
	MCD3	95743	11968	11968

Table 5: Statistics of datasets and splits used in our experiments.

A.2 Data Example for Composition Reward

Table 4 lists examples from four compositional generalization benchmarks, together with the primitive and compositional abstractions used to compute the compositional reward.

B Warmup

Accuracy The SFT warm-up is used as an initialization stage to stabilize early training and to provide basic instruction-following ability before applying GRPO optimization. For fair comparasion, both SFT and GRPO start from the same supervised warm-up checkpoint. From this shared checkpoint, we continue training with two different objectives: continued SFT and GRPO. Thus, the relevant comparison is warm-up SFT + continued SFT versus warm-up SFT + GRPO. We include the accuracy of the warm-up stage in Table 6.

Training Steps We use the same checkpoint selection procedure for both SFT and GRPO to ensure a fair comparison. Specifically, for each dataset and model, we evaluate checkpoints on the dev at regular training step intervals. Training is stopped when the dev accuracy does not improve for 3 consecutive evaluations. We select the checkpoint with the best dev accuracy for test evaluation. For the warm-up stage, the checkpoint

	SCAN		COGS	GeoQuery		CFQ			
	length	turn left		template	length	MCD1	MCD2	MCD3	Avg
Llama									
SFT	11.88	69.03	81.12	50.36	43.21	91.04	74.85	77.42	81.10
GRPO-Warmup	9.56	57.63	74.91	46.27	38.85	86.47	70.05	74.75	77.09
Qwen									
SFT	18.41	78.06	82.09	73.65	28.21	88.52	73.98	73.67	78.72
GRPO-Warmup	15.67	75.91	76.43	68.84	25.51	84.93	69.37	70.57	74.95

Table 6: Results of SFT and GRPO-Warmup on compositional generalization benchmarks across various settings.

Dataset	Llama SFT	Llama Warm-up	Llama GRPO	Qwen SFT	Qwen Warm-up	Qwen GRPO
SCAN-length	1.58k	1.06k	0.80k	1.52k	1.06k	0.73k
SCAN-left	2.72k	1.36k	1.12k	2.48k	1.36k	0.91k
COGS	2.26k	1.5k	0.9k	2.32k	1.5k	0.96k
GeoQuery-template	112	72	80	112	72	96
GeoQuery-length	112	72	96	112	72	96
CFQ-mcd1	8.95k	5.96k	3.04k	10.4k	5.96k	3.14k
CFQ-mcd2	8.56k	5.96k	4.32k	9.72k	5.96k	4.51k
CFQ-mcd3	8.67k	5.96k	2.88k	9.86k	5.96k	2.93k

Table 7: Training steps for Llama and Qwen under SFT, warm-up, and GRPO settings.

is selected using a fixed number of epochs, chosen to ensure that the model can produce valid task-formatted outputs and avoid degenerate rollouts (i.e., cases where most samples are incorrect), which would otherwise lead to unstable optimization due to sparse rewards. In practice, we use 2 epochs for SCAN, 2 epochs for COGS, 15 epochs for GeoQuery, and 3 epochs for CFQ. Detailed training configurations for each dataset and model are provided in Table 7.

C Implementation Details

Memory Usage We use H20 for training. In our setup, SFT requires approximately 42 GB per GPU, while GRPO requires approximately 64 GB per GPU. The higher memory usage of GRPO mainly comes from maintaining the rollout engine: we use vLLM to accelerate multi-trajectory sampling, which introduces additional GPU memory overhead compared with standard SFT.

Hyperparameter For the compositional reward weights, we conducted a small preliminary search on two representative development settings, SCAN-length and CFQ-MCD3, since they cover the productivity and systematicity settings discussed in the paper. We searched $\lambda_1 \in \{0.05, 0.1, 0.2, 0.4\}$ and $\lambda_2 \in \{0.05, 0.1, 0.2, 0.4\}$. The selected values were$\lambda_1 = 0.1$ and $\lambda_2 = 0.2$. After this preliminary selection, we fixed these weights for all datasets, model backbones, and splits.