Dissecting Reinforcement Learning: Mechanisms Behind Compositional Reasoning in LLMs

Gyeongwon James Kim

CMU-CS-26-107

May 2026

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Chenyan Xiong, Chair
Aditi Raghunathan

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

Keywords: Large Language Models, Reasoning Models, Reinforcement Learning, Compositional Generalization

For my family and friends who supported me through this journey.

Abstract

Recently, Large Reasoning Models (LRMs) have achieved impressive performance on a variety of reasoning tasks, including mathematics and code generation. Their long chains of reasoning enable scaling of inference-time computation, allowing them to solve increasingly complex problems. LRMs are typically post-trained from base models using supervised fine-tuning (SFT), reinforcement learning (RL), or a combination of both. RL is often hypothesized to be a key driver of reasoning ability, as it enables models to explore and discover new solutions. However, recent work suggests that RL may instead concentrate probability mass on existing solutions. The mechanisms by which RL leads to reasoning ability remain poorly understood, and modern RL pipelines bundle on-policy rollouts, variance normalization, KL divergence, entropy shaping, and other components into a single procedure, making it difficult to attribute gains to any specific cause. In this thesis, we study RL training mechanisms through the lens of compositional generalization—a key subskill of reasoning that involves combining atomic skills to solve more complex problems. We propose a unified two-axis framework that organizes SFT and RL methods along a *data* axis (off-policy to on-policy) and a *loss function* axis (positive-only to positive-plus-negative to GRPO). This framework places existing post-training algorithms as instances of a shared design space and enables controlled ablations of individual components. We instantiate this framework on a string-function composition task, where models must predict the output of composed string transformations of increasing depth. Our experiments yield a clear hierarchy among RL components. On the data axis, on-policy data alone is insufficient: positive-only on-policy SFT contracts response length and entropy and underperforms teacher-distilled off-policy SFT at higher composition levels. On the loss function axis, introducing negative gradients on top of on-policy training recovers a large fraction of the gap to GRPO, identifying the use of negative samples as the primary driver of compositional ability. Adding a group-mean baseline closes nearly all of the remaining gap, while the additional standard-deviation normalization in full GRPO contributes relatively little. These findings suggest that the bulk of GRPO's advantage on compositional generalization comes from the joint use of on-policy rollouts and baseline-stabilized negative gradients, rather than from group-wise variance normalization or other objective-level details. More broadly, effective post-training of LLMs benefits from a component-level view of training: practitioners can mix and match individual components to obtain most of RL's benefits, rather than treating RL as a monolithic improvement over SFT.

Acknowledgments

First and foremost, I would like to thank my advisor, Chenyan Xiong, for his invaluable guidance and support throughout the program. His insights shaped not only the direction of this thesis but also my growth as a researcher, and I am deeply grateful for his patience, encouragement, and mentorship.

I am also grateful to Aditi Raghunathan for serving as my second reader and for offering thoughtful feedback that strengthened this work. I would like to extend special thanks to Emmy Liu, who generously shared her time, technical expertise, and perspective. Many of the ideas and experiments in this thesis benefited directly from our discussions, and her guidance made this research process far more enjoyable.

Finally, I want to thank my friends and family. Their unwavering love, patience, and encouragement carried me through every stage of this journey, and this thesis would not have been possible without them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recent advances in Large Language Model (LLM) post-training have produced a new class of systems capable of solving problems that once seemed far beyond reach: competition mathematics, graduate-level science questions, and complex multi-step code generation. The defining ingredient is reinforcement learning with verifiable rewards (RLVR), exemplified by DeepSeek-R1 [6] and a wave of follow-up work. In these systems, a language model generates candidate solutions, a lightweight verifier checks correctness, and the model is updated to produce better solutions over time without human labels. As a result, the resulting models, Large Reasoning Models (LRMs), can produce multi-chain reasoning traces that allow them to solve increasingly complex problems.

Yet the mechanisms behind how RL leads to reasoning ability remain poorly understood. At least three explanations compete in the literature. The *discovery* view explains that RL allows the model to discover novel solutions by exploring the solution space and being rewarded accordingly [28]. On the other hand, the *sharpening* view holds that RL mostly concentrates probability mass on outputs already existing in the base model, improving single-sample accuracy without fundamentally expanding the model's capabilities [35]. Finally, the *chaining* view argues that RL implicitly teaches the model to chain multiple skills together, which is necessary to solve complex reasoning problems [23].

Furthermore, it is unclear which components of RL are responsible for the success of RL algorithms like GRPO over standard SFT. Some works argue that the key is the alignment between training data and the model's current distribution: because RL continuously generates rollouts from the evolving policy, it avoids the off-policy bias of static supervised datasets [2, 16]. Other works show that unlike SFT, which only reinforces correct demonstrations, RL explicitly penalizes incorrect outputs, driving the model to avoid failure modes [26]. Setlur et al. [23] shows that negative gradients are responsible for increasing the reasoning length, which is evidence for chaining. In practice, modern RL pipelines bundle all of these effects together with additional ingredients (KL regularization, variance normalization, entropy shaping), making it difficult to attribute gains to any single cause. *This paper addresses this gap by presenting a* ***unified framework*** *that effectively organizes existing post-training methods and isolate core components for analysis.*

Our approach decomposes training into two orthogonal axes and varies each independently. The first axis is **data**: we compare teacher-generated off-policy data, base-model bootstrapped

data, and fully on-policy rollouts, asking how the source of training trajectories affects compositional performance. The second axis is the **loss function**: starting from the SFT loss function (positive samples only), we progressively introduce negative gradient updates, a group mean baseline, and variance normalization, arriving at the full GRPO objective. By crossing these two axes we construct a grid of training methods that interpolates between standard off-policy SFT and standard GRPO, isolating the contribution of each component and creating a *unified framework* to understand existing post-training paradigms.

We investigate training mechanisms through the lens of *compositional generalization*: the ability to chain multiple known atomic skills together to solve more complex problems. Crucially, the gap between SFT and RL is especially pronounced here: SFT-trained models plateau sharply as composition depth increases, while RL-trained models continue to improve, creating a clean and measurable performance separation [3, 34]. This is important, as compositional generalization is a key ingredient in general reasoning ability [20, 35]. A model solving a complex problem may need to decompose the task into multiple sub-tasks and solve each sub-task sequentially. We use the string-function composition task introduced by Yuan et al. [34], where the model must predict the output of composed string transformations of increasing depth.

Our experimental results reveal a clear hierarchy among the components of RL. On the data axis, on-policy data alone is not sufficient to close the gap to GRPO: on-policy SFT in fact underperforms teacher-distilled off-policy SFT, because training on positive-only on-policy rollouts contracts response length and entropy, suppressing the long chain-of-thought trajectories needed for higher-level composition. On the loss axis, introducing negative gradients on top of on-policy training (the POS+NEG setting) recovers a substantial portion of the gap to GRPO, identifying the use of negative samples as the key driver of compositional ability. Adding a group-mean baseline (REINFORCE+Baseline) recovers nearly all of the remaining gap, indicating that the baseline is the key to stabilizing training. Together, these results show that the bulk of GRPO's advantage on compositional generalization comes from the combination of on-policy rollouts and negative gradients with a baseline, rather than from group-wise variance normalization or other objective-level details.

This paper makes the following contributions:

1. **A unified decomposition framework.** We organize SFT and RL variants under a common framework parameterized by data policy (off-policy to on-policy) and loss function (positive-only to positive+negative to GRPO). This framework clarifies the design space and enables controlled ablations.
2. **On-policy data is necessary but not sufficient**: On-policy data is necessary to support the negative-gradient signal, since negative samples must be drawn from the model's own distribution to be informative. However, on its own, on-policy data does not close the gap to GRPO: positive-only on-policy SFT contracts response length and entropy and underperforms teacher-distilled off-policy SFT on higher composition levels.
3. **Use of negative gradients is a key component**: Loss functions that use negative samples to train greatly outperform SFT, which only uses positive samples. In addition, techniques that normalize the reward signal relative to the task's difficulty show the best performance over methods that naively train on positive and negative samples.

Chapter 2

Related Work

2.1 Training Large Reasoning Models

The development of reasoning-capable LLMs has been shaped by a progression of post-training techniques. Early work introduced bootstrapping via self-generated reasoning traces [36], and reinforcement learning from human feedback established RL as a practical post-training tool [18]. More recently, DeepSeek-R1 [6] demonstrated that RL training directly from a base model (zero-RL) can produce strong reasoning behavior, while also finding that a supervised cold-start further stabilizes training and improves downstream performance. Follow-up systems such as DAPO [33] introduced objective modifications—clip-higher, entropy-aware rewards, and token-mean normalization—to prevent collapse during prolonged RL training. SimpleRL-Zoo [37] investigated how problem difficulty, output length, and data diversity interact during zero-RL, finding that problem difficulty is critical and that long CoT SFT can suppress exploration. ProRL [13] studies the regime of extended RL training across many task categories and shows that maintaining exploration throughout requires explicit diversity incentives.

The training dynamics of RL post-training are increasingly well characterized. Scaling laws for RL post-training follow power-law relationships with compute and data [25]. Work using synthetic reasoning tasks identifies an "edge of competence" effect: RL works best when the base model already has partial competence on a task, and the interplay between pretraining coverage, mid-training distribution, and RL optimization target is a key determinant of compositional generalization [39]. A separate line of work identifies specific behavioral primitives—verification, subgoal decomposition, backtracking, backward chaining—that make a model more compatible with RL training [11].

Despite this progress, most production reasoning systems bundle curriculum schedules, data mixing, reward shaping, and objective modifications into a single pipeline, making it difficult to attribute improvements to specific mechanisms. This thesis addresses this gap by presenting a unified framework for analyzing existing post-training methods.

2.2 SFT versus Reinforcement Learning

Several studies argue that SFT and RL produce qualitatively different distributional changes. Under a controlled arithmetic task, Chu et al. [4] find that SFT improves in-distribution accuracy via memorization while RL generalizes to out-of-distribution settings by exploiting underlying model capabilities, and that SFT can stabilize format for downstream RL. A complementary trajectory-level analysis shows that RL reduces the proportion of incorrect completions while SFT increases the proportion of correct ones: RL compresses wrong trajectories, SFT expands right ones [15]. These findings suggest the two methods differ not just in their objective but in which region of the output distribution they reshape.

A second perspective emphasizes the formal relationship between SFT and RL. SFT on curated data can be viewed as maximizing a lower bound on a sparse-reward RL objective, and importance-weighted SFT tightens this bound and behaves more like RL [22]. A unified policy-gradient framework shows that SFT, DPO-style objectives, and RL share a common structure, with key distinctions being the data generating policy, the weighting over outcomes, and the stabilization constraint [14]. Combining SFT-style and RL-style signals in a single training stage can be mutually beneficial, with entropy serving as an informative diagnostic of training dynamics [10].

RL post-training also mitigates catastrophic forgetting. Chen et al. [2] show that RL forgets prior capabilities significantly less than SFT at matched task performance, attributing this advantage primarily to on-policy data collection rather than to KL regularization or advantage weighting. Lai et al. [12] investigates this in a continual post-training setting. More broadly, Wu et al. [31] characterize SFT as inducing a directional drift away from prior capabilities that RL counteracts, rather than discovering genuinely new solutions.

While prior works have focused on studying the effects of SFT and RL separately, this thesis takes a more fine-grained approach by analyzing the component-wise contributions of SFT and RL in a unified framework.

2.3 Exploration and Sharpening

An influential debate concerns whether RL primarily *sharpens* existing behaviors or whether it genuinely *explores* new ones. The sharpening view holds that RL shifts probability mass toward high-reward completions that are already latent in the base model, improving pass@1 while potentially reducing pass@k diversity. Evidence for this view comes from pass@k analyses showing that base models eventually surpass RL-trained models at large k [35], from theoretical characterizations of RLVR as support-constrained optimization that progressively narrows the exploration frontier [30], and from observations of entropy collapse during long training runs [43].

The exploration view holds that RL can discover qualitatively new strategies not well represented in the base model. SimpleRL-Zoo [37] argues that exploration exceeds sharpening in accounting for RL gains. Methods that preserve exploration throughout training include entropy regularization [5], representation-space novelty bonuses [27], and experience replay strategies that revisit valuable earlier trajectories to prevent mode collapse [9, 38, 40].

2.4 Extrapolation, Chaining, and Compositional Generalization

Compositional generalization, the ability to recombine known operations in novel arrangements, is a stringent test of systematic reasoning. LLMs often exhibit compositional deficiencies. Despite strong performance on individual skills, models often fail when those skills must be chained [21, 42]. The OMEGA benchmark operationalizes this by evaluating models that master atomic skills on compositional and transformative combinations, finding substantial failure rates for state-of-the-art models [24]. Theoretical analysis shows that data coverage required for multi-hop compositional generalization can grow super-linearly with task diversity, while models that leverage compositional structure or function reuse can generalize more efficiently [1].

The evidence that RL helps with compositional generalization is growing. Using a synthetic string-function task, which we adopt in this thesis, Yuan et al. [34] show that RL enables models to compose $f(g(x))$ from separately learned f and g, while off-policy SFT on the composed task fails to generalize. Motwani et al. [17] demonstrate that RL with a curriculum of increasing compositional depth improves substantially even at high pass@k, suggesting genuine capability acquisition rather than pure sharpening. Composable chain-of-thought data is critical for enabling models to chain operations systematically, complementing RL supervision [32]. Work on pre/mid/post-training interactions using a synthetic task identifies an edge of competence in the base model as a prerequisite for RL-driven compositional generalization, and shows that mid-training can bridge the pretraining and post-training distributions to improve this precondition [39].

2.5 On-Policy Training

A recurring theme in the SFT-versus-RL literature is the role of on-policy data. Ming et al. [16] show that the performance gap between SFT and RL is largely attributable to the off-policy versus on-policy distinction rather than the form of the loss: converting static SFT data into token-level on-policy training consistently outperforms standard SFT. Taken to its logical conclusion, removing the KL regularization and group-wise normalization from a GRPO objective reduces the algorithm to on-policy SFT while still achieving competitive performance and substantially shorter reasoning traces [41]. Anchored SFT [44] adds lightweight KL regularization to on-policy training to prevent distributional drift while preserving the tightness of the RL lower bound.

These findings motivate the on-policy axis of our decomposition, but they also leave open a residual question: on-policy SFT does not generally close the full gap to RL, suggesting that other components such as negative gradients provide additional independent signal. This thesis directly tests the impact of on-policy data for compositional generalization.

2.6 Role of Negative Gradients in Training

Standard SFT provides only positive gradients: it increases the likelihood of correct trajectories without penalizing incorrect ones. RL objectives that use advantage weighting implicitly apply negative updates by suppressing below-average completions, but the causal contribution of this negative signal has received limited direct study.

A first challenge is that naive use of negative samples can produce failure modes. In DPO, the standard contrastive loss can *reduce* the absolute likelihood of preferred responses as long as the relative gap between preferred and dispreferred completions increases [19]. An analogous issue arises in GRPO: Deng et al. [7] identify *Lazy Likelihood Displacement* (LLD), where the likelihood of correct responses marginally increases or even decreases during training because all tokens in incorrect responses are penalized with equal strength regardless of their contribution. Their proposed fix, NTHR, down-weights penalties on tokens that cause LLD by using co-occurring correct responses as anchors. Crucially, they find that negative gradients are not inherently harmful and that the failure mode stems from undifferentiated uniform penalization rather than the negative signal itself.

A second line of evidence shows that negative samples carry genuine learning signal for generalization. Tian et al. [26] show that incorporating negative trajectories into SFT, specifically those with valid intermediate reasoning steps but incorrect final answers, yields substantial out-of-domain generalization gains over positive-only training. Furthermore, Di et al. [8] find that high-quality negatives exhibiting expected format and structural coherence consistently outperform randomly chosen or naively filtered incorrect responses.

Taken together, these results establish that the negative gradient is a meaningful and distinct training signal whose effective exploitation requires attention to which tokens are penalized and what quality of negatives are used. In this thesis, we provide complementary evidence by directly measuring the causal contribution of the negative gradient component in GRPO on compositional generalization.

Chapter 3

Compositional Generalization

3.1 Task Definition

Compositional generalization is the ability to compose multiple atomic skills to solve a harder problem. Hence, benchmarks for compositional ability define the notion of an atomic skill and construct more complex tasks by composing these skills. Generally, we can measure the difficulty of the task by the amount of composition required to solve it. To make this more concrete, we study a toy task where the goal is to predict the output of functions that map strings to strings [34].

3.2 Toy Task: String Functions

We study compositional generalization using the string-function setting as proposed in Yuan et al. [34]. Each string function takes as input one or more Python strings and outputs a new string. The model receives definitions of simple atomic functions and must predict outputs of composed programs, where the difficulty is controlled by composition depth. A level-i task requires composing i operations. Figure 3.1 shows an overview of the string function task. The dataset contains 25 unique string functions, split into 13 in the train set and 12 in the eval set.

We follow the training protocol proposed in Yuan et al. [34]. For any pretrained model, we *initialize* the model by training it on level-1 tasks with standard Supervised Fine-Tuning. This ensures that the model masters the atomic tasks. We call the resulting checkpoint **Base**, which is used as the starting point for compositional training.

- **Compositional Training:** To train for composition, we train on level-2 tasks starting from the **Base** model. We use level-2 tasks since they are the lowest difficulty tasks that requires composition. At this training stage, we vary the training algorithm across the data and loss function axes, and study their effect on the compositional ability of the final model.
- **Evaluation:** We measure performance on a separate evaluation set on tasks from level 1 to level 8. This allows us to see how performance extrapolates to higher composition levels.

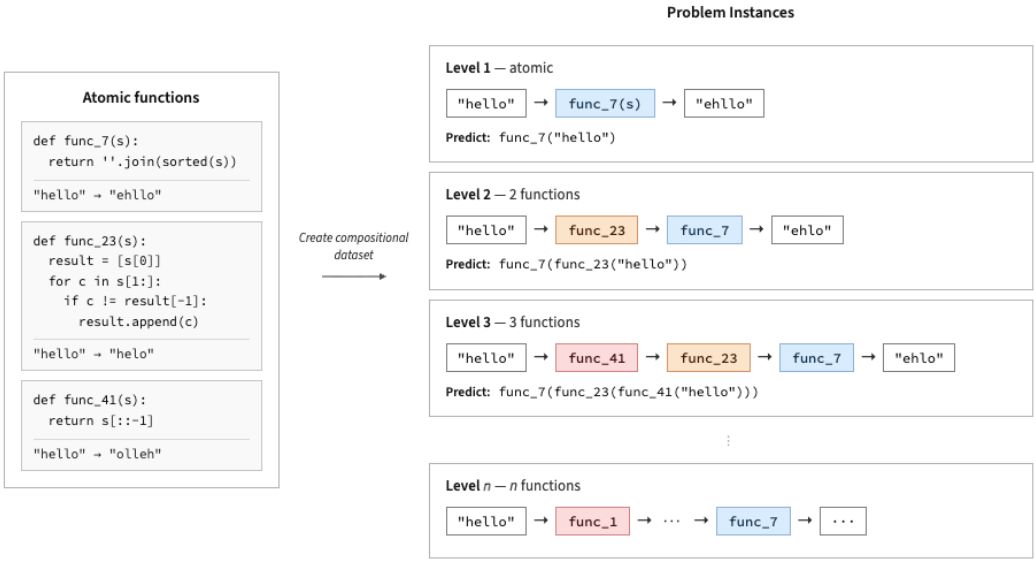


Figure 3.1: Overview of the string function task

Chapter 4

Decomposing Training Mechanisms

4.1 A Unified Two-Axis Framework

	Positive Samples Only	**Positive and Negative Samples**
Off-policy	SFT	DPO
On-policy	On-policy SFT	GRPO

Table 4.1: The Unified Two-Axis Framework

Our goal is to decompose the training mechanisms across various SFT and RL methods and organize them into a unified framework. As a starting point, we identify the two core factors that differentiate SFT with GRPO: on-policy data and use of negative samples for gradient updates. Table 4.1 classifies known post-training methods into a two-axis grid, which allows us to place popular post-training algorithms like SFT, DPO, and GRPO. We generalize the two components into the following axis:

- **Data**: this involves the way we source and select data that will be used for training.
- **Loss function**: this involves the way we update the model parameters based on the data.

4.2 Data Axis

One common distinction between methods in RL is whether the data is on-policy or off-policy. On-policy data is generated by the current model being trained, while off-policy data is generated by a previous reference model, humans, or teacher models. To study the role of the data source, we formalize the following settings:

- **Teacher**: teacher-generated trajectories
- **Bootstrap**: bootstrapped trajectories from the initial model checkpoint

- **Iterative**: bootstrapped trajectories from the previous model checkpoint. We can control the number of model checkpoints, k, to use for bootstrapping. If $k = 1$, it is the same as Bootstrap-SFT. If $k =$ number of training steps, it is the same as On-policy SFT.
- **On-policy**: on-policy trajectories throughout training

Data Source	Classification
Teacher	Off-policy
Bootstrap	Off-policy
Iterative(k)	Off-policy
On-policy	On-policy

Table 4.2: Classification of Data Source methods

In accordance with the literature, we consider the **Teacher**, **Bootstrap**, and **Iterative** settings as Off-policy methods since they make gradient updates based on data from older models or teacher models. However, we can still smoothly transition from Off-policy to On-policy methods by increasing the number of model checkpoints used in the **Iterative** setting.

4.3 Loss Function Axis

Our key insight is that we can view *loss functions as methods that apply different weights to positive and negative samples.* Table 4.3 summarizes how different loss functions weight positive and negative samples. Note that this understanding applies for tasks with binary reward ($r \in \{1, 0\}$).

Use of Negative Samples	Loss Function	Positive Weight	Negative Weight
No	SFT (4.1)	1	0
	GRPOMask (4.5)	$(1-\mu)/\sigma$	0
Yes	POS+NEG (4.2)	1	-1
	REINFORCE+ (4.3)	$1-\mu$	$-\mu$
	GRPO (4.4)	$(1-\mu)/\sigma$	$-\mu/\sigma$

Table 4.3: Comparison of loss functions by weights used for positive and negative samples.

First, we start with the standard loss function for Supervised Fine-Tuning (SFT).

$$\ell_{\text{SFT}}(\theta) = -\mathbb{E}_{(x,y)}\left[\frac{1}{T}\sum_{t=1}^{T}\log \pi_\theta(y_t|x, y_{<t})\right]$$

For consistency with RL training procedures, we consider SFT as generating multiple rollouts for the same problem instance and filtering the positive samples for training. Let G denote the

number of generated samples per problem. Furthermore, we can rewrite the following term:

$$\log \pi_\theta(y|x) = \Sigma_{t=1}^{T} \log \pi_\theta(y_t|x, y_{<t})$$

Then, the loss simplifies to

$$\ell_{\text{SFT}}(\theta) = -\mathbb{E}_{(x,\{y\}^{G^+})} \left[\frac{1}{G} \sum_{i=1}^{G} \frac{1}{T} \log \pi_\theta(y_i|x) \right]$$

where G^+ denotes the set of positive samples.

Since SFT only uses positive samples, we can rewrite the loss as:

$$\ell_{\text{SFT}}(\theta) = -\mathbb{E}_{(x,\{y\}^{G})} \left[\frac{1}{G} \sum_{i=1}^{G} \mathbf{1}[r_i = 1] \cdot \frac{1}{T} \log \pi_\theta(y_i|x) \right] \tag{4.1}$$

This derivation demonstrates that SFT is a special case where positive samples have weight 1 and negative sample have weight 0.

As a simple way to incorporate negative samples, we have the following loss function that increases likelihood of positive samples and decreases likelihood of negative samples:

$$\ell_{\text{POS+NEG}}(\theta) = -\mathbb{E}_{(x,\{y\}^{G})} \left[\frac{1}{G} \sum_{i=1}^{G} \mathbf{1}[r_i = 1] \cdot \log \pi_\theta(y_i|x) - \mathbf{1}[r_i = 0] \cdot \log \pi_\theta(y_i|x) \right] \tag{4.2}$$

This is essentially equivalent to REINFORCE [29] with rewards $r_i \in \{1, -1\}$ instead of $\{1, 0\}$. In other words, the loss function simply places $+1$ weight on positive samples and -1 weight on negative samples.

Next, we can further add a baseline to the REINFORCE loss where μ is the mean reward of the group:

$$\ell_{\text{REINFORCE+}}(\theta) = -\mathbb{E}_{(x,\{y\}^{G})} \left[\frac{1}{G} \sum_{i=1}^{G} (r_i - \mu) \cdot \log \pi_\theta(y_i|x) \right] \tag{4.3}$$

This takes into account the relationship of samples within G. If most samples in the group are correct, then the mean is high and the weight on each positive sample is low. On the other hand, if only a few samples are correct, the mean is low and the weight on positive samples are high. The same is true for negative samples. Hence, we can see loss 4.3 as placing $1 - \mu$ weight on positive samples and $-\mu$ weight on negative samples.

Finally, we use standard deviation σ as a normalization factor to get to GRPO loss.

$$\ell_{\text{GRPO}}(\theta) = -\mathbb{E}_{(x,\{y\}^{G})} \left[\frac{1}{G} \sum_{i=1}^{G} \left(\frac{r_i - \mu}{\sigma} \right) \cdot \log \pi_\theta(y_i|x) \right] \tag{4.4}$$

For simplicity, we do not consider clipping and KL-divergence penalty. In this simplified version, we see that GRPO loss weighs the samples using a group-normalized advantage: $\frac{1-\mu}{\sigma}$ for positive samples and $\frac{-\mu}{\sigma}$ for negative samples.

Another setting used to isolate the effect of negative gradients is to apply a mask to the negative gradients in GRPO [23].

$$\ell_{\text{GRPOMask}}(\theta) = -\mathbb{E}_{(x,\{y\}^{G})} \left[\frac{1}{G} \sum_{i=1}^{G} \mathbf{1}[r_i = 1] \cdot \left(\frac{r_i - \mu}{\sigma} \right) \cdot \log \pi_\theta(y_i|x) \right] \tag{4.5}$$

Chapter 5

Experiments

5.1 Setup

We use Llama-3.1-8B-Instruct as the backbone model. As noted in section 3.2, we first train this model on level-1 tasks to ensure that the model has acquired the atomic skills. This setup stage uses Rejection Fine-Tuning (RFT) on 117k samples. We call the resulting checkpoint **Base**, which is used as the starting point for our main training methods. During **Compositional Training**, we train the **Base** model with various training methods across the data and loss function axes as described in section 4.1. By evaluating the performance of the model on level-i tasks with $i \geq 2$, we can measure the compositional generalization ability of the model.

5.1.1 Matching Compute

To conduct a fair comparision between methods, we utilize a matched compute setting, where compute is measured by the number of samples used in the training process. For On-policy methods, let b be the batch size and T be the total training steps. For each problem instance, we rollout G samples (this is also the group size in GRPO). Then, the total number of samples used is bTG. For Off-policy methods, we use the teacher model or checkpoint model to generate G solutions for bT problem instances, matching the total sample count. One thing to note is that when we use SFT loss, we only use positive samples. However, we still measure the compute by the total number of samples *before* filtering out the negatives. By default, we use a batch size of $b = 16$ and $T = 1200$ training steps, resulting in 19200 unique problem instances. We use group size of $G = 16$, resulting in total sample count of 307200.

5.2 Baselines

In order to understand the effect of different components, we establish lower and upper bounds for performance. Namely, the standard off-policy Bootstrap SFT method establishes the baseline lower bound performance. On the other hand, standard on-policy GRPO establishes our gold-standard upper bound in performance. Figure 5.1 shows the performance gap, which we hope to close by ablating various components.

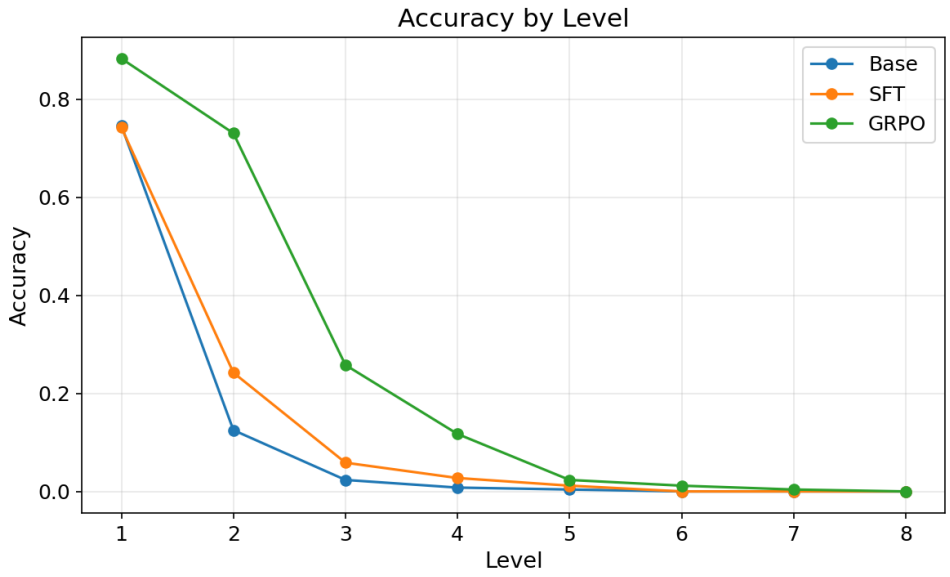


Figure 5.1: Accuracy comparing the Base model to 2 training methods: SFT and GRPO

5.3 Data Axis

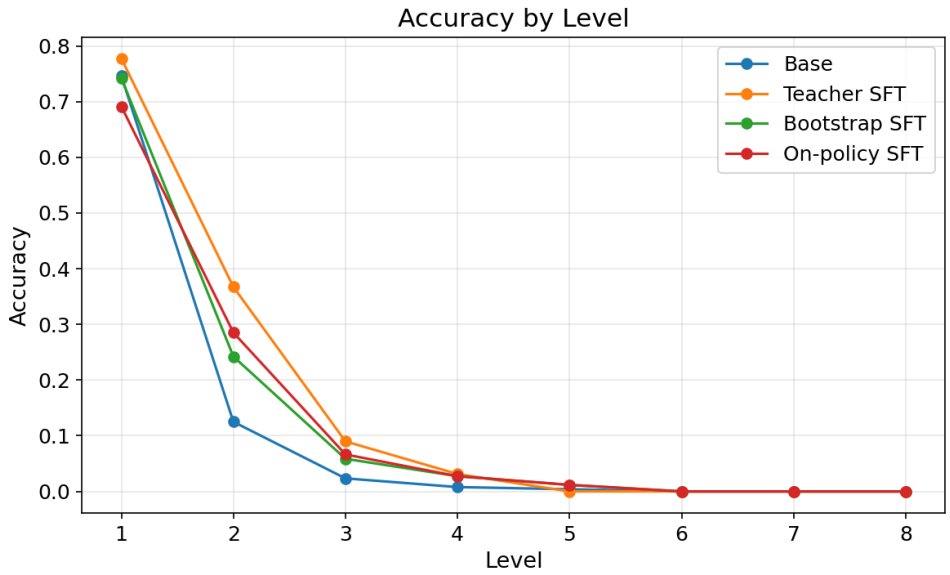


Figure 5.2: Accuracy comparing Off-policy and On-policy SFT methods

Figure 5.2 shows the accuracy of the model after SFT with different data sources. All methods meaningfully improve performance over the **Base** model. In addition, using teacher-generated data and on-policy data outperform bootstrapped data from the initial model checkpoint. Surprisingly, Teacher SFT, an off-policy data method, outperforms on-policy SFT. We

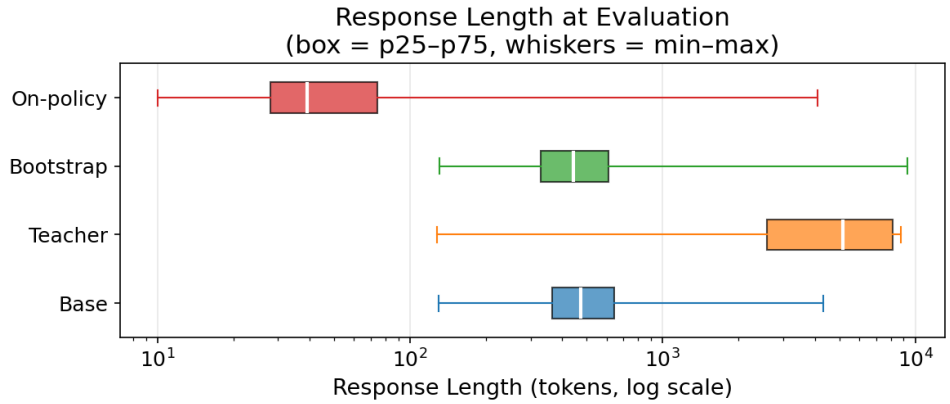


Figure 5.3: Response length distribution on evaluation set for models trained with SFT on different data sources.

explain this result using Figure 5.3, which shows the response length distribution of the models after training. As expected, the distribution in the Bootstrap setting is similar to the **Base** model since the data used during training comes from the initial model checkpoint. However, we see 2 very distinct behaviors when comparing On-policy vs Teacher SFT settings. The teacher data comes from the On-policy GRPO checkpoint, which leads the model to imitate the long chain-of-thought trajectories. On the contrary, in the on-policy setting, the model learns to contract the response, leading to a decrease in overall response length throughout training. As a result, the teacher method generalizes better to unseen samples.

5.4 Loss Function

We report the evaluation accuracy of different loss functions under the On-policy setting in Figure 5.4. We find that the gap between using SFT loss and GRPO loss is large. By adding negative samples (POS+NEG setting), we see that we can recover a significant chunk of the performance. However, there is still a performance gap, suggesting that naively adding negative gradients is not enough. Only after adding a baseline (REINFORCE+Baseline setting), we achieve very close to GRPO performance. This suggests that group-mean baseline is a key factor while standard-deviation normalization has relatively smaller impact.

Figure 5.5 shows the average response length during training. Crucially, we see that SFT loss (only training on positive samples) contracts response length. On the other hand, by training on negative samples as well, response length increases during training. We see that in GRPO and REINFORCE+Baseline settings, the response length increases steadily throughout training, achieving best performance around 1000 training steps. Interestingly, POS+NEG setting shows more instability in training as response length rapidly increases in the first stage, then gradually returns back to the Base model's levels. A possible explanation for this is that in the POS+NEG setting, the total magnitude of positive weights and negative weights are unbalanced. For example, if a group has much more positive samples than negative samples, the magnitude of positive

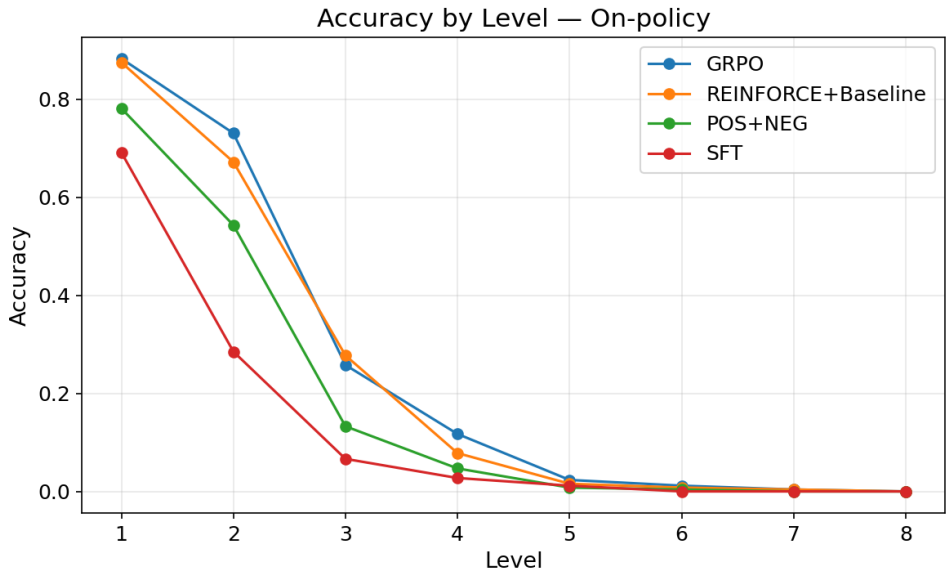
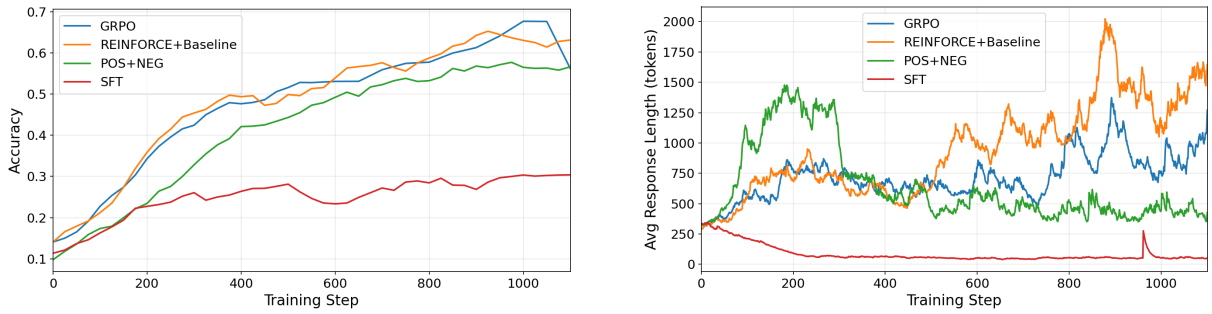


Figure 5.4: Accuracy comparing different loss functions under On-policy setting



(a) Level 2 Eval Accuracy

(b) Average Response length

Figure 5.5: Accuracy and Average response length over training steps under On-policy data setting

gradient updates is much larger than that of negative gradient updates. By adding a baseline (REINFORCE+Baseline, GRPO settings), we balance the magnitude of gradient updates from positive and negative samples. As a result, POS+NEG setting is more sensitive to the difficulty of tasks while GRPO and REINFORCE+Baseline settings are more stable, eventually achieving greater performance.

Chapter 6

Future Work

Our findings establish a controlled decomposition of RL components on the string-function task with Llama-3.1-8B-Instruct. We propose two natural extensions that will strengthen these conclusions and establish their broader relevance.

6.1 Transfer to General Reasoning Tasks

The string-function setting was chosen because it isolates compositional generalization in a controlled way: atomic skills are well-defined, composition depth is precisely controllable, and outputs are exactly verifiable. A central open question is whether the component-wise findings established here transfer to the broader reasoning tasks that are more relevant to real world tasks such as mathematics, code generation, and general-purpose multi-step reasoning.

We propose two complementary directions for testing transfer. First, we can ask whether compositional ability acquired through our task functions as a transferable sub-skill. Concretely, we would compare a base model against a model trained on the string-function task and evaluate both on standard reasoning benchmarks such as MATH-500, AMC, AIME, OlympiadBench, code reasoning suites, and general reasoning benchmarks. If the compositionally trained model shows non-trivial improvements on these benchmarks despite the obvious surface-level mismatch, it would provide evidence that compositional generalization is a transferable primitive of reasoning rather than a task-specific artifact.

Second, we can test whether the training-method conclusions themselves generalize. Our finding that on-policy rollouts plus negative gradients with a baseline recover most of GRPO's advantage was established in a setting where reward is binary and verification is exact. Real reasoning tasks typically have noisier reward signals, longer chain-of-thought trajectories, and more complex failure modes. Re-running the unified two-axis decomposition on math and code benchmarks would test whether the same component hierarchy holds, and would identify which components, if any, are specific to the controlled compositional setting versus broadly applicable to reasoning post-training.

6.2 Generality across Model Families and Sizes

Our main experiments use Llama-3.1-8B-Instruct as the backbone. While this provides a clean and reproducible setting, it leaves open the question of whether our conclusions are specific to this model's pretraining distribution, scale, or instruction-tuning recipe. The role of negative gradients, in particular, may interact with how well-calibrated the base model is and with the entropy of its initial output distribution.

To test generality, we propose to repeat the decomposition across several axes. Along the size axis, the Qwen3 family (0.6B, 1.7B, 4B, 8B, 14B) provides a controlled scaling sweep within a single pretraining recipe, enabling us to test whether the contribution of each RL component is stable across model scale or whether smaller models rely on different components than larger ones. Along the family axis, comparing Llama and Qwen models at matched scale would test whether the component hierarchy depends on pretraining data and instruction-tuning choices.

Together, these two extensions would convert the controlled findings of this thesis into more actionable guidance for practitioners designing post-training pipelines for reasoning models.

Chapter 7

Conclusion

This thesis set out to answer a question that has become increasingly central to LLM post-training: *what makes reinforcement learning effective for compositional reasoning, and which of its many ingredients are responsible?* Modern RL pipelines such as GRPO bundle on-policy rollouts, negative gradients, group-mean baselines, variance normalization, KL regularization, and curriculum schedules into a single training procedure, making it difficult to attribute observed gains to any specific mechanism. Prior work has illuminated individual components in isolation, but no controlled study has placed them in a common framework that allows for direct head-to-head comparison.

To address this, we proposed a unified two-axis decomposition of post-training methods. The *data* axis ranges from teacher-distilled and bootstrapped off-policy data to fully on-policy rollouts, while the *loss* axis ranges from positive-only SFT through positive-plus-negative gradients, REINFORCE with a group-mean baseline, and finally to the full GRPO objective. This grid places SFT, on-policy SFT, DPO, and GRPO as instances of a shared design space and exposes the components that distinguish them. We instantiated this framework on the string-function composition task, which provides a clean, controllable test of compositional generalization where the SFT-versus-RL gap is large and easily measurable.

Our experiments yield a clear hierarchy among RL components. On the data axis, on-policy data alone is insufficient: positive-only on-policy SFT contracts response length and entropy and underperforms teacher-distilled off-policy SFT at higher composition levels. On the loss axis, introducing negative gradients on top of on-policy training recovers a large fraction of the gap to GRPO, identifying the use of negative samples as the primary driver of compositional ability. Adding a group-mean baseline closes nearly all of the remaining gap, while the additional standard-deviation normalization in full GRPO contributes relatively little. Taken together, the bulk of GRPO's advantage on compositional generalization comes from the combination of on-policy rollouts and negative gradients with a stabilizing baseline, rather than from group-wise variance normalization or other objective-level details.

These findings carry two broader implications. First, they support a *component-level* view of post-training: rather than treating RL as a monolithic improvement over SFT, practitioners can mix and match individual components (negative gradients, baselines, on-policy data) to obtain most of RL's benefits at lower implementation and compute cost. Second, they sharpen the ongoing debate over why RL works for reasoning. Our results are consistent with the view

that negative gradients enable the model to suppress unproductive reasoning patterns rather than merely sharpen existing ones, and that on-policy data primarily serves to make this negative signal informative rather than to drive learning on its own.

Several limitations remain. Our conclusions are established on a single controlled toy task and a single backbone model, and the binary, exactly verifiable reward of the string-function setting is simpler than rewards encountered in math and code benchmarks. Extending the decomposition to general reasoning tasks and to additional model families and sizes, as outlined in the Future Work chapter, is necessary to determine which parts of the component hierarchy are universal and which are task- or model-specific.

This thesis contributes a unified framework, a controlled set of ablations, and a concrete attribution of GRPO's advantage to the joint use of on-policy rollouts and baseline-stabilized negative gradients. We hope these results help shift post-training research toward principled, component-wise analysis, and ultimately toward more transparent and efficient training of reasoning-capable language models.

Bibliography

[1] Amirhesam Abedsoltan, Huaqing Zhang, Kaiyue Wen, Hongzhou Lin, Jingzhao Zhang, and Mikhail Belkin. Task Generalization With AutoRegressive Compositional Structure: Can Learning From D Tasks Generalize to D^{T} Tasks?, June 2025. URL http://arxiv.org/abs/2502.08991. arXiv:2502.08991 [cs]. 2.4

[2] Howard Chen, Noam Razin, Karthik Narasimhan, and Danqi Chen. Retaining by Doing: The Role of On-Policy Data in Mitigating Forgetting, December 2025. URL http://arxiv.org/abs/2510.18874. arXiv:2510.18874 [cs]. 1, 2.2

[3] Sitao Cheng, Xunjian Yin, Ruiwen Zhou, Yuxuan Li, Xinyi Wang, Liangming Pan, William Yang Wang, and Victor Zhong. From Atomic to Composite: Reinforcement Learning Enables Generalization in Complementary Reasoning, December 2025. URL http://arxiv.org/abs/2512.01970. arXiv:2512.01970 [cs]. 1

[4] Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V. Le, Sergey Levine, and Yi Ma. SFT Memorizes, RL Generalizes: A Comparative Study of Foundation Model Post-training, May 2025. URL http://arxiv.org/abs/2501.17161. arXiv:2501.17161 [cs]. 2.2

[5] Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, Zhiyuan Liu, Hao Peng, Lei Bai, Wanli Ouyang, Yu Cheng, Bowen Zhou, and Ning Ding. The Entropy Mechanism of Reinforcement Learning for Reasoning Language Models, May 2025. URL http://arxiv.org/abs/2505.22617. arXiv:2505.22617 [cs]. 2.3

[6] DeepSeek-AI, Daya Guo, and Yang. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning, January 2025. URL http://arxiv.org/abs/2501.12948. arXiv:2501.12948 [cs]. 1, 2.1

[7] Wenlong Deng, Yi Ren, Muchen Li, Danica J. Sutherland, Xiaoxiao Li, and Christos Thrampoulidis. On the Effect of Negative Gradient in Group Relative Deep Reinforcement Optimization, May 2025. URL http://arxiv.org/abs/2505.18830. arXiv:2505.18830 [cs]. 2.6

[8] Zixiang Di, Jinyi Han, Shuo Zhang, Ying Liao, Zhi Li, Xiaofeng Ji, Yongqi Wang, Zheming Yang, Ming Gao, Bingdong Li, and Jie Wang. Not All Negative Samples Are Equal: LLMs Learn Better from Plausible Reasoning, February 2026. URL http://arxiv.org/abs/2602.03516. arXiv:2602.03516 [cs] version: 2. 2.6

[9] Shihan Dou, Muling Wu, Jingwen Xu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang.

Improving RL Exploration for LLM Reasoning through Retrospective Replay, July 2025. URL http://arxiv.org/abs/2504.14363. arXiv:2504.14363 [cs]. 2.3

[10] Yuqian Fu, Tinghong Chen, Jiajun Chai, Xihuai Wang, Songjun Tu, Guojun Yin, Wei Lin, Qichao Zhang, Yuanheng Zhu, and Dongbin Zhao. SRFT: A Single-Stage Method with Supervised and Reinforcement Fine-Tuning for Reasoning, June 2025. URL http://arxiv.org/abs/2506.19767. arXiv:2506.19767 [cs]. 2.2

[11] Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D. Goodman. Cognitive Behaviors that Enable Self-Improving Reasoners, or, Four Habits of Highly Effective STaRs, August 2025. URL http://arxiv.org/abs/2503.01307. arXiv:2503.01307 [cs]. 2.1

[12] Song Lai, Haohan Zhao, Rong Feng, Changyi Ma, Wenzhuo Liu, Hongbo Zhao, Xi Lin, Dong Yi, Qingfu Zhang, Hongbin Liu, Gaofeng Meng, and Fei Zhu. Reinforcement Fine-Tuning Naturally Mitigates Forgetting in Continual Post-Training, January 2026. URL http://arxiv.org/abs/2507.05386. arXiv:2507.05386 [cs]. 2.2

[13] Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. ProRL: Prolonged Reinforcement Learning Expands Reasoning Boundaries in Large Language Models, May 2025. URL http://arxiv.org/abs/2505.24864. arXiv:2505.24864 [cs]. 2.1

[14] Xingtai Lv, Yuxin Zuo, Youbang Sun, Hongyi Liu, Yuntian Wei, Zhekai Chen, Xuekai Zhu, Kaiyan Zhang, Bingning Wang, Ning Ding, and Bowen Zhou. Towards a Unified View of Large Language Model Post-Training, January 2026. URL http://arxiv.org/abs/2509.04419. arXiv:2509.04419 [cs]. 2.2

[15] Kohsei Matsutani, Shota Takashiro, Gouki Minegishi, Takeshi Kojima, Yusuke Iwasawa, and Yutaka Matsuo. RL Squeezes, SFT Expands: A Comparative Study of Reasoning LLMs, September 2025. URL http://arxiv.org/abs/2509.21128. arXiv:2509.21128 [cs]. 2.2

[16] Rui Ming, Haoyuan Wu, Shoubo Hu, Zhuolun He, and Bei Yu. One-Token Rollout: Guiding Supervised Fine-Tuning of LLMs with Policy Gradient, January 2026. URL http://arxiv.org/abs/2509.26313. arXiv:2509.26313 [cs]. 1, 2.5

[17] Sumeet Ramesh Motwani, Alesia Ivanova, Ziyang Cai, Philip Torr, Riashat Islam, Shital Shah, Christian Schroeder de Witt, and Charles London. h1: Bootstrapping LLMs to Reason over Longer Horizons via Reinforcement Learning. 2025. doi: 10.48550/ARXIV.2510.07312. URL https://arxiv.org/abs/2510.07312. Version Number: 2. 2.4

[18] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, P. Welinder, P. Christiano, Jan Leike, and Ryan J. Lowe. Training language models to follow instructions with human feedback. *ArXiv*, March 2022. URL https://www.semanticscholar.org/paper/Training-language-models-to-follow-instructions-Ouyang-Wu/d766bffc357127e0dc86dd69561d5aeb520d6f4c. 2.1

[19] Arka Pal, Deep Karkhanis, Samuel Dooley, Manley Roberts, Siddartha Naidu, and Colin White. Smaug: Fixing Failure Modes of Preference Optimisation with DPO-Positive, July 2024. URL http://arxiv.org/abs/2402.13228. arXiv:2402.13228 [cs]. 2.6

[20] Simon Park, Simran Kaur, and Sanjeev Arora. How Does RL Post-training Induce Skill Composition? A Case Study on Countdown, December 2025. URL http://arxiv.org/abs/2512.01775. arXiv:2512.01775 [cs]. 1

[21] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. Measuring and Narrowing the Compositionality Gap in Language Models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.378. URL https://aclanthology.org/2023.findings-emnlp.378/. 2.4

[22] Chongli Qin and Jost Tobias Springenberg. Supervised Fine Tuning on Curated Data is Reinforcement Learning (and can be improved), September 2025. URL http://arxiv.org/abs/2507.12856. arXiv:2507.12856 [cs]. 2.2

[23] Amrith Setlur, Matthew Y. R. Yang, Charlie Snell, Jeremy Greer, Ian Wu, Virginia Smith, Max Simchowitz, and Aviral Kumar. e3: Learning to Explore Enables Extrapolation of Test-Time Compute for LLMs, June 2025. URL http://arxiv.org/abs/2506.09026. arXiv:2506.09026 [cs]. 1, 4.3

[24] Yiyou Sun, Shawn Hu, Georgia Zhou, Ken Zheng, Hannaneh Hajishirzi, Nouha Dziri, and Dawn Song. OMEGA: Can LLMs Reason Outside the Box in Math? Evaluating Exploratory, Compositional, and Transformative Generalization, June 2025. URL http://arxiv.org/abs/2506.18880. arXiv:2506.18880 [cs]. 2.4

[25] Zelin Tan, Hejia Geng, Xiaohang Yu, Mulei Zhang, Guancheng Wan, Yifan Zhou, Qiang He, Xiangyuan Xue, Heng Zhou, Yutao Fan, Zhongzhi Li, Zaibin Zhang, Guibin Zhang, Chen Zhang, Zhenfei Yin, Philip Torr, and Lei Bai. Scaling Behaviors of LLM Reinforcement Learning Post-Training: An Empirical Study in Mathematical Reasoning, December 2025. URL http://arxiv.org/abs/2509.25300. arXiv:2509.25300 [cs]. 2.1

[26] Xueyun Tian, Minghua Ma, Bingbing Xu, Nuoyan Lyu, Wei Li, Heng Dong, Zheng Chu, Yuanzhuo Wang, and Huawei Shen. Learning from Mistakes: Negative Reasoning Samples Enhance Out-of-Domain Generalization, January 2026. URL http://arxiv.org/abs/2601.04992. arXiv:2601.04992 [cs] version: 1. 1, 2.6

[27] Jens Tuyls, Dylan J. Foster, Akshay Krishnamurthy, and Jordan T. Ash. Representation-Based Exploration for Language Models: From Test-Time to Post-Training. 2025. doi: 10.48550/ARXIV.2510.11686. URL https://arxiv.org/abs/2510.11686. Version Number: 1. 2.3

[28] Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. Reinforcement Learning with Verifiable Rewards Implicitly Incentivizes Correct Reasoning in Base LLMs, October 2025. URL http://arxiv.org/abs/2506.14245. arXiv:2506.14245 [cs]. 1

[29] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist re-

inforcement learning. *Mach. Learn.*, 8(3–4):229–256, May 1992. ISSN 0885-6125. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696. 4.3

[30] Fang Wu, Weihao Xuan, Ximing Lu, Mingjie Liu, Yi Dong, Zaid Harchaoui, and Yejin Choi. The Invisible Leash: Why RLVR May or May Not Escape Its Origin, January 2026. URL http://arxiv.org/abs/2507.14843. arXiv:2507.14843 [cs]. 2.3

[31] Yongliang Wu, Yizhou Zhou, Zhou Ziheng, Yingzhe Peng, Xinyu Ye, Xinting Hu, Wenbo Zhu, Lu Qi, Ming-Hsuan Yang, and Xu Yang. On the Generalization of SFT: A Reinforcement Learning Perspective with Reward Rectification, October 2025. URL http://arxiv.org/abs/2508.05629. arXiv:2508.05629 [cs]. 2.2

[32] Fangcong Yin, Zeyu Leo Liu, Liu Leqi, Xi Ye, and Greg Durrett. Learning Composable Chains-of-Thought, May 2025. URL http://arxiv.org/abs/2505.22635. arXiv:2505.22635 [cs]. 2.4

[33] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. 2025. doi: 10.48550/ARXIV.2503.14476. URL https://arxiv.org/abs/2503.14476. Version Number: 2. 2.1

[34] Lifan Yuan, Weize Chen, Yuchen Zhang, Ganqu Cui, Hanbin Wang, Ziming You, Ning Ding, Zhiyuan Liu, Maosong Sun, and Hao Peng. From $f(x)$ and $g(x)$ to $f(g(x))$: LLMs Learn New Skills in RL by Composing Old Ones, December 2025. URL http://arxiv.org/abs/2509.25123. arXiv:2509.25123 [cs]. 1, 2.4, 3.1, 3.2, 3.2

[35] Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does Reinforcement Learning Really Incentivize Reasoning Capacity in LLMs Beyond the Base Model?, May 2025. URL http://arxiv.org/abs/2504.13837. arXiv:2504.13837 [cs]. 1, 2.3

[36] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping Reasoning With Reasoning, May 2022. URL http://arxiv.org/abs/2203.14465. arXiv:2203.14465 [cs]. 2.1

[37] Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. SimpleRL-Zoo: Investigating and Taming Zero Reinforcement Learning for Open Base Models in the Wild, May 2025. URL http://arxiv.org/abs/2503.18892. arXiv:2503.18892 [cs]. 2.1, 2.3

[38] Runzhe Zhan, Yafu Li, Zhi Wang, Xiaoye Qu, Dongrui Liu, Jing Shao, Derek F. Wong, and Yu Cheng. ExGRPO: Learning to Reason from Experience, October 2025. URL http://arxiv.org/abs/2510.02245. arXiv:2510.02245 [cs]. 2.3

[39] Charlie Zhang, Graham Neubig, and Xiang Yue. On the Interplay of Pre-Training, Mid-Training, and RL on Reasoning Language Models, December 2025. URL http://arxiv.org/abs/2512.07783. arXiv:2512.07783 [cs]. 2.1, 2.4

[40] Hongzhi Zhang, Jia Fu, Jingyuan Zhang, Kai Fu, Qi Wang, Fuzheng Zhang, and Guorui Zhou. RLEP: Reinforcement Learning with Experience Replay for LLM Reasoning, July 2025. URL http://arxiv.org/abs/2507.07451. arXiv:2507.07451 [cs]. 2.3

[41] Anhao Zhao, Ziyang Chen, Junlong Tong, Yingqi Fan, Fanghua Ye, Shuhao Li, Yunpu Ma, Wenjie Li, and Xiaoyu Shen. On-Policy Supervised Fine-Tuning for Efficient Reasoning, February 2026. URL http://arxiv.org/abs/2602.13407. arXiv:2602.13407 [cs]. 2.5

[42] Jun Zhao, Jingqi Tong, Yurong Mou, Ming Zhang, Qi Zhang, and Xuanjing Huang. Exploring the Compositional Deficiency of Large Language Models in Mathematical Reasoning, October 2024. URL http://arxiv.org/abs/2405.06680. arXiv:2405.06680 [cs] version: 4. 2.4

[43] Rosie Zhao, Alexandru Meterez, Sham Kakade, Cengiz Pehlevan, Samy Jelassi, and Eran Malach. Echo Chamber: RL Post-training Amplifies Behaviors Learned in Pretraining, August 2025. URL http://arxiv.org/abs/2504.07912. arXiv:2504.07912 [cs]. 2.3

[44] He Zhu, Junyou Su, Peng Lai, Ren Ma, Wenjia Zhang, Linyi Yang, and Guanhua Chen. Anchored Supervised Fine-Tuning, February 2026. URL http://arxiv.org/abs/2509.23753. arXiv:2509.23753 [cs]. 2.5