PARIF: Pushing the Pareto Frontier of Instruction Following and Reasoning with Curriculum Reinforcement Learning

Rongchuan Mu[1*], **Zexin Wang**[1*], **Qianyu Wang**[1*],
Minghua Ma[1], **Zekun Wang**[1], **Ming Liu**[1,2†], **Bing Qin**[1,2]
[1]Harbin Institute of Technology, Harbin, China
[2]Pengcheng Laboratory, Shenzhen, China.
{rcmu,zxwang,qywang,mhma,zkwang,mliu,qinb}@ir.hit.edu.cn
Project Page: https://github.com/qianyu0415/PARIF

Abstract

Large Reasoning Models (LRMs) excel at complex problem-solving but frequently overlook specific instruction constraints. Existing alignment methods struggle to balance general reasoning with instruction-following (IF), hindered by dependency on teacher models, reward hacking, and reasoning-answer inconsistencies. We propose **PARIF**, a two-stage curriculum learning framework based on Reinforcement Learning from Verifiable Rewards (RLVR) to enhance both IF and general reasoning capabilities. The framework employs a *correctness proxy* across different stages to mitigate reward hacking. Stage I employs a *dynamic weighting strategy* simultaneously to optimize the model's reasoning paradigm regarding constraints. Stage II introduces *Decoupled-GRPO*, which builds upon the first stage to enhance the logical consistency between the reasoning process and the final answer, enabling the model to better leverage its optimized reasoning paradigm. To support the framework, we curate 26,000 high-quality instructions featuring diverse constraints. Extensive experiments demonstrate PARIF's effectiveness: our 7B model achieves a remarkable **21.25%** relative average improvement to the original model across six representative IF tasks, while our 8B model outperforms leading models like DeepSeek-V3 on these IF tasks, effectively pushing the Pareto frontier of instruction following and reasoning for models of comparable scale. We open-source our code and models to facilitate future research.

1 Introduction

Recent studies (Dubey et al., 2024; Liu et al., 2024a; Yang et al., 2025; Anthropic, 2025) have established Instruction Following (IF) as a foundational capability for aligning Large Language Models (LLMs) with human intent. However, the emergence of Large Reasoning Models (LRMs) has introduced a critical trade-off: while they have unlocked advanced capabilities in handling complex reasoning tasks (Guo et al., 2025a; Team et al., 2025), this cognitive depth is frequently accompanied by **instruction-following degradation** (Li et al., 2025; Fu et al., 2025). Consequently, even state-of-the-art models exhibit suboptimal performance in general constraint adherence (He et al., 2025a; Li et al., 2025).

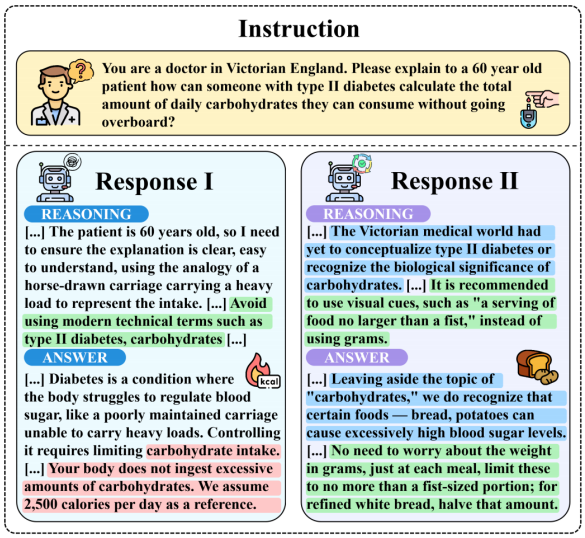


Figure 1: Examples of logical (in)consistency between the reasoning and the answer. In the left response, the model fails to adhere to its own reasoning, leading to a logical discrepancy. In the right response, the answer is highly consistent with the reasoning.

Building upon the success of Reinforcement Learning with Verifiable Rewards (RLVR) (Guo et al., 2025a; Qian et al., 2025), recent works have adapted this paradigm to enhance the instruction-following capabilities of LRMs. They verify the compliance of model responses with specific constraints and devise corresponding reward mechanisms to bolster IF capabilities. However, these methods fail to effectively eliminate the instruction-

[*]Equal Contribution.
[†]Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 24753–24783
July 2-7, 2026

following degradation due to three critical challenges: (1) **Teacher Dependence**: they rely on teacher supervision, which not only requires costly data distillation but also caps performance within the teacher's distribution (Chang et al., 2025; Qin et al., 2025a; Wang et al., 2025); (2) **Reward Hacking**: verifiable rewards can be exploited with shortcut behaviors or verifier-specific patterns without genuine constraint understanding, especially without explicit anti-hacking designs (Lambert et al., 2024a; Ren et al., 2025); (3) **Reasoning-Answer inconsistency**: existing methods rely solely on outcome rewards, ignoring the inconsistency between reasoning and the final answer (Figure 1). This oversight can lead to constraint violations, indicating that current approaches remain suboptimal (Pyatkin et al., 2025; Guo et al., 2025b). We thus ask: **How can we design a teacher-free RLVR framework that is robust to reward hacking and enhances reasoning-answer consistency, thereby pushing outward the Pareto frontier between instruction following and reasoning performance?**

We address this by proposing **PARIF**, a two-stage curriculum learning framework based on RLVR, which does not rely on teacher models and performs self-exploration solely through reinforcement learning. The framework introduces a **Correctness Proxy**, which provides a unified interpretation for correctness judgment across different stages and data types, effectively preventing reward hacking. Furthermore, we propose **Decoupled-GRPO** to align the reasoning and answer by using credit assignment strategies (Sutton, 1984; Arjona-Medina et al., 2019). This algorithm reallocates outcome rewards to the reasoning process as a whole based on logical consistency with the answer segment, measured by posterior probability.

Specifically, in Stage I, we optimize the model's reasoning paradigm regarding constraints to stimulate deep thinking by employing Correctness Proxy and Dynamic Constraint Weighting. In Stage II, we employ Decoupled-GRPO to enhance the logical consistency between the reasoning process and the answer, enabling the model to better leverage the reasoning paradigm optimized in Stage I. To support the framework, we collect a wide range of constraint templates and construct a high-quality dataset containing 26,000 instructions. We conduct extensive experiments on three reasoning models ranging from 1.7B to 8B parameters. Ultimately, across six IF benchmarks and six general reasoning benchmarks, our 7B model obtains relative performance gains of 21.25% and 5.80%, and the 8B model obtains relative gains of 5.47% and 1.58%, effectively pushing the Pareto frontier. Our main contributions are summarized as follows:

- We construct a high-quality verifiable dataset of 26,000 instructions with diverse constraints, built through a scalable and automated process requiring less manual annotation.
- We propose PARIF, a two-stage RLVR-based curriculum framework that integrates our novel Decoupled-GRPO algorithm. By optimizing reasoning processes and final answers in a decoupled manner, it enhances instruction following and logical consistency while mitigating reward hacking and preserving generative diversity.
- We conduct extensive experiments and analysis, demonstrating that our models achieve SOTA performance across multiple IF benchmarks and deliver measurable improvements in reasoning-intensive tasks, pushing the Pareto frontier.

2 Preliminaries

2.1 Problem Setup

Let π_θ denote a reasoning policy model parameterized by θ. Given a query $x \sim \mathcal{D}$, the reasoning policy's output o can be decomposed into a reasoning process z followed by an answer segment y. Thus, the joint probability can be factorized as:

$$\pi_\theta(o|x) = \pi_\theta(z, y|x) = \underbrace{\pi_\theta(z|x)}_{\text{Reasoning}} \cdot \underbrace{\pi_\theta(y|x, z)}_{\text{Answering}}. \tag{1}$$

The RLVR objective is to maximize the expected correctness of the final response:

$$\max_\theta \mathbb{E}_{x\sim\mathcal{D}} \left[\mathbb{E}_{(z,y)\sim\pi_\theta(\cdot|x)} \mathcal{V}(y, x)\right], \tag{2}$$

where $\mathcal{V}(y, x) \in \{0, 1\}$ is a deterministic verification function. It evaluates whether the answer segment y aligns with the ground-truth (if available) or satisfies predefined criteria in the absence of an explicit reference.

2.2 Outcome Supervision GRPO

Our framework is based on the GRPO algorithm with outcome-based rewards (Shao et al., 2024). To optimize the reasoning policy more effectively, the framework adopts the following improvements proposed in DAPO (Yu et al., 2025): Clip-Higher and Token-Level Policy Gradient Loss. Specifically,

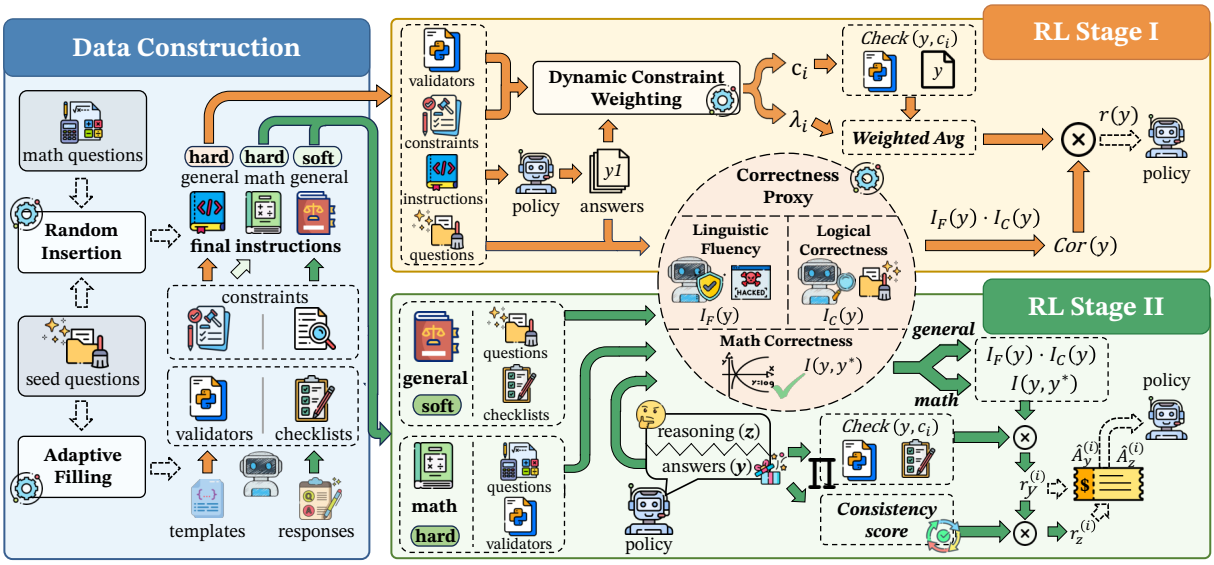


Figure 2: The PARIF framework. **Left:** The data construction pipeline. We synthesize final instructions by integrating constraints into seed and math questions. **Upper right:** Overview of Stage I. This stage employs dynamic constraint weighting (λ_i) to overcome optimization bottlenecks and a correctness proxy to mitigate reward hacking. **Lower right:** Overview of Stage II. Building upon Stage I, this stage computes the consistency score between the reasoning and answers of the policy models, enabling the decoupled optimization of both components.

for a query x, we sample G responses using the policy model π_θ. Then, the advantage of the i-th response is calculated by normalizing the group-level rewards $\{r_i\}_{i=1}^G$:

$$\hat{A}_i = \frac{r_i - \text{mean}(\{r_i\}_{i=1}^G)}{\text{std}(\{r_i\}_{i=1}^G)}. \tag{3}$$

After computing the advantages, we optimize the following objective:

$$\begin{aligned}\mathcal{J}(\theta) = \mathbb{E}_{[x\sim\mathcal{D},\{o_i\}_{i=1}^G\sim\pi_{\theta_{\text{old}}}(\cdot|x)]} \frac{1}{\sum_{i=1}^G |o_i|} \\ \sum_{i=1}^{G}\sum_{t=1}^{|o_i|}\Bigg\{\min\Big[\rho_{i,t}\hat{A}_i, \\ \text{clip}(\rho_{i,t}, 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}})\hat{A}_i\Big] \\ -\beta D_{\text{KL}}[\pi_\theta\|\pi_{\text{ref}}]\Bigg\},\end{aligned} \tag{4}$$

where $\rho_{i,t}$ denotes the importance sampling ratio, $|o_i|$ denotes the total number of tokens in the i-th output, ϵ_{low} and ϵ_{high} denote the decoupled lower and upper clipping bounds, D_{KL} represents the KL divergence, and β is the corresponding coefficient.

3 Method

As illustrated in Figure 2, our framework consists of data construction (§3.1) followed by a two-stage curriculum: Stage I optimizes the model's reasoning paradigm regarding constraints (§3.2), while Stage II enhances the logical consistency between the reasoning process and the answer, further boosting capabilities across diverse tasks (§3.3).

3.1 Data Construction

Unlike distillation, we avoid synthesizing or sampling reference responses. Results demonstrate that our framework achieves superior performance (see Table 13). Data construction consists of the following three main stages.

Seed Collection. We collect 52k English-only seed questions from diverse sources (shareAI, 2023; Es, 2023; Zheng et al., 2023; Taori et al., 2023; Zhao et al., 2024), filtering out multi-turn dialogues, mathematical questions, and programming-related questions.

Constraint Construction. Constraints are categorized into *hard constraints* (verifiable by code) and *soft constraints* (requiring LLM evaluation). We compile 108 hard constraint templates with several placeholders, and organize them into six distinct categories. Corresponding Python verification functions accept placeholders as arguments, thereby eliminating the need to generate a dedicated verification function for each individual problem, enabling scalable data generation. For soft

constraints, we adopt the approach proposed in VerIF (Peng et al., 2025), which facilitates the generation of natural constraints with comprehensive verification checklists.

Data Synthesis. We generate constraint combinations with the Cartesian product, filter out conflicting sets, and integrate them into the seed questions. During this process, we adopt the following strategies: (1) **Prompt-Level Conflict Detection.** We verify whether the constraint combination conflicts with the logic of the original seed prompt. (2) **Adaptive Template Filling.** We dynamically instantiate constraint templates based on the context of the query. (3) **Randomized Position Insertion.** We randomize the order of the original question and the added constraints to generate the final instruction. We preserve the semantic integrity of original questions through string concatenation, followed by LLM-based filtering to ensure fluency. These strategies mitigate logical conflicts and alleviate positional bias.

For more details on data construction, the scope of instruction following, and the constraint taxonomy, see Appendix A.2.

3.2 Stage I: RLVR for basic IF Improvement

This stage enhances basic IF by focusing exclusively on hard constraints, which provide clearer reward signals and place more emphasis on the instruction adherence than soft constraints. To mitigate reward hacking, we introduce dynamic constraint weighting and a correctness proxy.

Dynamic Constraint Weighting. Static weighting , whether uniform or pre-defined (Wang et al., 2025), frequently yields optimization bottlenecks by biasing the model toward trivial or disproportionately weighted constraints. Thus we dynamically adjust each constraint's weight based on the model's evolving proficiency.

Specifically, given an instruction x containing m hard constraints $\{c_1, c_2, \ldots, c_m\}$, we sample G responses from the model. The dynamic weight λ_i for c_i is calculated as:

$$\lambda_i = 1 - \frac{1-\delta}{G} \sum_{j=1}^{G} \text{Check}(y_j, c_i), \tag{5}$$

where y_j denotes the answer segment in the j-th output, and $\text{Check}(y, c)$ is a binary verifier. The hyperparameter δ (set to 0.7 in this work) serves as a maintenance margin. It ensures that even fully mastered constraints contribute a baseline reward signal, preventing the reward vanishing problem.

Correctness Proxy. To address the issue of models over-optimization (Guo et al., 2025b), this module assesses the overall quality of the generated response. Specifically, we posit that a high-quality response must satisfy two criteria: (1) **Linguistic Fluency**, ensuring the content is coherent and readable; and (2) **Logical Correctness**, ensuring the response correctly addresses the query. Formally, we define the correctness proxy as:

$$\text{Cor}(y) = \mathbb{I}_F(y) \cdot \mathbb{I}_C(y), \tag{6}$$

where $\mathbb{I}_F(y) \in \{0, 1\}$ indicates the response is free of potential hacking patterns based on a pre-defined checklist, and $\mathbb{I}_C(y) \in [0, 1]$ measures whether the answer correctly solves the original question. The final reward is formulated as:

$$r(y) = \text{Cor}(y) \cdot \frac{1}{m} \sum_{i=1}^{m} \lambda_i \cdot \text{Check}(y, c_i). \tag{7}$$

Once the rewards are obtained, the advantages are computed by Eq. 3, and the policy model is updated by optimizing Eq. 4. More details are provided in Appendix A.3.

3.3 Stage II: Decoupled-GRPO

This stage restores general reasoning and enhances real-world IF generalization by introducing math questions (Luo et al., 2025) and soft constraints. We first compute the output reward as follows:

$$r(x, y) = \text{Cor}(x, y) \cdot \prod_{i=1}^{m} \text{Check}(y, c_i), \tag{8}$$

$$\text{Cor}(x, y) = \begin{cases} \mathbb{I}(y, y^*), & x \in \mathcal{D}_{\text{math}} \\ \mathbb{I}_F(y) \cdot \mathbb{I}_C(y), & \text{otherwise}, \end{cases} \tag{9}$$

where y^* denotes the ground truth. We extend the correctness proxy to incorporate mathematical ground-truth while maintaining Stage I's strategy for non-math data. For soft constraints, we employ LLMs to verify whether they have been successfully satisfied. The high precision of mathematical verification further reduces noise and mitigates reward hacking.

Decoupled GRPO Algorithm. We feed the model's full response (comprising the reasoning process and the answer segment) back for an additional forward pass to compute the consistency score as follows:

$$\text{Cons}(o) = \frac{\sum_{t_i \in y} p_i}{|y|}, \tag{10}$$

where p_i denotes the probability of the i-th token in the answer segment, and $|y|$ represents the length of the answer segment. This score essentially serves as the posterior probability of the answer segment conditioned on the reasoning process, reflecting the logical consistency between the two. A higher probability indicates that the model considers the answer to be a natural extension of the current reasoning process. Furthermore, this metric also quantifies the level of certainty in the model's answer segment. As illustrated in Figure 3, a high-quality reasoning process is generally associated with higher predictive certainty. Consequently, the consistency score also functions as an implicit proxy that reflects the quality of the underlying reasoning process. For further analysis on the rationality of the consistency score design, see Appendix A.3.

We thus define separate rewards for the reasoning and answer components as $r^{(i)}_{\text{reason}} = r(x, y_i) \cdot \text{Cons}(o_i)$ and $r^{(i)}_{\text{answer}} = r(x, y_i)$, respectively. This reward formulation can be viewed as a more general form of the outcome-based reward paradigm. Instead of employing a fixed unit coefficient, we utilize $\text{Cons}(o)$ as a coefficient to allocate the reward from the answer to the reasoning process and explicitly optimize this coefficient. Unlike calibration-based approaches, we do not explicitly incentivize the model to reduce its consistency scores. Instead, our objective is to primarily optimize the binary outcome reward $r(x, y)$, while further refining consistency.

To address the discrepancy in reward distributions (where r_{reason} lies in a continuous space $[0, 1]$ while r_{answer} belongs to a discrete set $\{0, 1\}$), we adopt the strategy from PopArt (Hessel et al., 2019) and calculate the advantages for the two components separately:

$$\hat{A}^{(i)}_{\text{reason}} = \frac{r^{(i)}_{\text{reason}} - \text{mean}(\{r^{(i)}_{\text{reason}}\}_{i=1}^{G})}{\text{std}(\{r^{(i)}_{\text{reason}}\}_{i=1}^{G})}, \tag{11}$$

$$\hat{A}^{(i)}_{\text{answer}} = \frac{r^{(i)}_{\text{answer}} - \text{mean}(\{r^{(i)}_{\text{answer}}\}_{i=1}^{G})}{\text{std}(\{r^{(i)}_{\text{answer}}\}_{i=1}^{G})}. \tag{12}$$

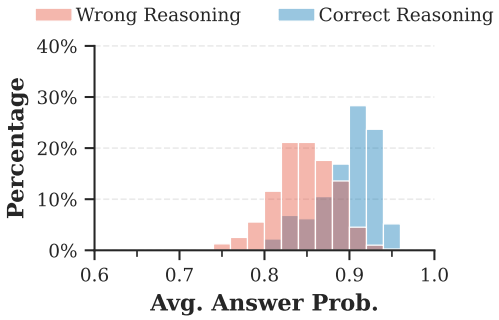


Figure 3: Probability distribution of answer segments on AIME25. The x-axis denotes the average token probability of the answer segment, and the y-axis represents the proportion within correct/wrong reasoning processes.

Following Eq. 4, we define the specific objectives for the reasoning and answer components, respectively. The total objective function is thus formulated as:

$$\begin{aligned}\mathcal{J}_{\text{Total}}(\theta) = \mathcal{J}_{\text{Reason}}(\theta) + \mathcal{J}_{\text{Answer}}(\theta) \\ + \alpha \mathcal{H}_{Reason}(\pi_\theta),\end{aligned} \tag{13}$$

where $\mathcal{H}_{\text{reason}}$ denotes the entropy of tokens within the reasoning process, and α is the entropy coefficient. We explicitly incorporate this entropy term to maintain the model's exploratory capacity during the reasoning process. We use the baseline model derived from Stage I as the reference model. The detailed algorithm is provided in Appendix A.3.

4 Experiments

In this section, we evaluate our framework on multiple IF benchmarks and general benchmarks to investigate the following research questions (further analyses are provided in Appendix A.5):

- **RQ1:** Does our framework effectively push the Pareto frontier between general reasoning capabilities and IF capabilities? (§ 4.2)
- **RQ2:** Does our framework rely on specific family of verifier models? (§ 4.3)
- **RQ3:** Does enhancing consistency offer additional advantages over the direct use of outcome-based rewards? (§ 4.3)
- **RQ4:** Does enhancing consistency compromise model's generative diversity? (§ 4.4)
- **RQ5:** Does our framework exacerbate the overconfidence phenomenon in LRMs? (§ 4.4)

Model	IFEval		FollowBench		IFBench		InfoBench	CFBench			Multi-IF			Avg.
	Pr. (S.)	Pr. (L.)	HSR	SSR	Pr. (S.)	Pr. (L.)	Overall	CSR	ISR	PSR	Step1	Step2	Step3	
Large Baseline Models (≥ 70B)														
Qwen2.5-72B-Inst.	85.10	87.36	62.58	67.29	36.05	39.46	83.33	87	61	72	83.70	71.50	60.90	69.02
GPT-4o	83.40	87.80	64.84	69.85	31.00	35.70	82.58	88	64	73	84.42	73.57	64.68	69.45
DeepSeek-V3	79.93	84.94	73.28	75.89	31.63	40.82	86.80	91	70	78	84.50	76.03	65.90	72.21
Doubao-1.5-pro	84.76	88.29	62.44	67.93	28.57	33.67	82.31	88	66	74	88.26	80.84	73.99	70.70
Instruction-Tuned Baseline Models														
Qwen2.5-7B-Inst.	33.09	39.78	34.83	45.81	13.61	18.03	68.09	57	23	31	35.17	23.32	17.47	33.86
Mistral-Crab-7B-DPO	49.70	57.70	49.44	56.49	18.37	22.79	77.91	67	28	40	48.15	36.36	28.90	44.68
Llama-3.1-Tulu-3-8B	77.27	79.74	62.16	67.03	23.81	26.27	81.69	81	47	58	80.91	67.47	55.18	62.12
Llama-3.1-Tulu-3-8B-VerIF	80.30	82.53	58.55	64.34	22.79	24.15	79.47	81	48	58	81.89	68.66	58.31	62.15
Reasoning Models (1.7B)														
Qwen3-1.7B	70.06	74.49	51.61	58.18	22.49	28.57	81.51	82	51	62	72.58	62.57	52.45	59.19
PARIF-Stage I	71.35	75.23	52.07	58.63	**27.55**	**32.99**	82.76	82	52	63	73.40	63.39	53.51	60.61 (+2.40%)
PARIF-Stage II	**74.31**	**78.00**	**54.21**	**59.90**	26.19	30.95	**83.03**	**83**	**54**	**64**	**73.84**	**64.53**	**53.57**	**61.50 (+3.90%)**
Reasoning Models (7B)														
R1-Distill-Qwen-7B	57.99	63.38	43.82	53.68	10.88	14.97	75.73	74	40	50	59.91	46.27	34.49	48.09
R1-Distill-Qwen-7B-RAIF	60.78	66.54	**51.76**	57.84	15.31	19.73	79.78	77	44	55	68.80	54.23	41.97	53.29 (+10.81%)
R1-Distill-Qwen-7B-VerIF	**71.93**	**75.46**	50.77	58.96	23.81	26.53	79.64	79	46	57	**73.93**	60.25	48.21	57.81 (+20.21%)
PARIF-Stage I	68.96	71.56	47.83	57.26	22.45	25.85	79.60	79	46	56	71.89	59.58	47.94	56.46 (+17.40%)
PARIF-Stage II	70.70	74.72	51.20	**59.02**	**24.49**	**27.89**	**79.88**	**80**	**49**	**58**	73.68	**60.83**	**48.61**	**58.31 (+21.25%)**
Reasoning Models (8B)														
Qwen3-8B	83.73	86.87	62.04	65.22	31.01	35.77	85.82	**89**	66	75	87.33	79.66	71.76	70.71
Qwen3-8B-IFD	85.50	88.48	60.34	63.90	26.19	32.65	**86.80**	**89**	**67**	75	88.29	82.50	75.58	70.86 (+0.21%)
Qwen3-8B-LightIF	89.41	91.26	64.04	67.92	**40.14**	43.20	85.56	88	64	74	89.82	83.47	76.39	73.63 (+4.13%)
PARIF-Stage I	88.85	91.08	63.44	67.11	35.03	40.82	85.13	**89**	**67**	**76**	89.56	81.50	74.56	73.01 (+3.25%)
PARIF-Stage II	**89.78**	**92.57**	**66.06**	**69.16**	39.12	**44.56**	86.69	**89**	**67**	75	**90.50**	**83.55**	**76.43**	**74.58 (+5.47%)**

Table 1: Main results (%) on IF benchmarks. Pr.: prompt level; S./L.: Strict/Loose metrics; HSR/SSR: Hard/Soft Satisfaction Rate; CSR: Constraint Satisfaction Rate; ISR: Instruction Satisfaction Rate; PSR: Prompt Satisfaction Rate; IFD: IFDecorator method. For reasoning models, we remove the reasoning process and evaluate using only the answer segment. Best/2nd best are **bolded**/underlined.

4.1 Experimental Setup

Models & Baselines. We conduct experiments using three LRMs: Qwen3-1.7B, Qwen3-8B,[1] (Yang et al., 2025), and DeepSeek-R1-Distill-Qwen-7B (Guo et al., 2025a). We use Qwen3-235B-A22B-Instruct-2507 for data synthesis, and use Qwen3-Coder (Qwen Team, 2025) to generate Python verification functions. For the specific implementation, we employ Qwen3-32B as the default LLM verifier. For comparison, we evaluate the following methods that are based on RLVR: VerIF (Peng et al., 2025), Light-IF (Wang et al., 2025), RAIF (Qin et al., 2025b), and IFDECORATOR (Guo et al., 2025b). We also evaluate various larger models, including GPT-4o (Hurst et al., 2024), Qwen2.5-72B-Instruct (Team, 2024), DeepSeek-V3 (DeepSeek-AI, 2024), Doubao-1.5-pro (Team, 2025), and instruct models, including Llama-3.1-Tulu-3-8B-VerIF (Peng et al., 2025), Llama-3.1-Tulu-3-8B (Lambert et al., 2024a), Qwen2.5-7B-Instruct (Team, 2024), and Mistral-Crab-7B-DPO (Qi et al., 2025).

Evaluation Benchmarks. We evaluate our models on following representative IF benchmarks: IFEval (Zhou et al., 2023), FollowBench (Jiang et al., 2024), InFoBench (Qin et al., 2024a), and IFBench (Pyatkin et al., 2025). To evaluate the generalization ability of our models, we additionally select the following benchmarks, which have a completely different distribution from our training data: CFBench (Zhang et al., 2025), a Chinese comprehensive constraints following benchmark; and Multi-IF (He et al., 2024b), a multi-turn IF benchmark. Furthermore, to evaluate the general reasoning capabilities of our models, we select the following benchmarks: Math-500 (Lightman et al., 2023), AIME24 (Maxwell-Jia, 2024), AIME25 (math ai, 2025), MMLU-Pro (Wang et al., 2024b), WritingBench (Wu et al., 2025), and LiveCodeBench v5 (Jain et al., 2024). For all evaluations using LLM-as-a-judge, we adopt GPT-4o as the judge. More details on experimental settings and benchmarks are provided in Appendix A.4.

4.2 Main Results

Table 1 and Table 10 (in Appendix A.4) present the main results for IF and general benchmarks, respectively. Further details are provided in Appendix A.4. We draw the following key observations: (1) **Absolute IF Performance Improvements.** Our framework consistently delivers per-

[1]During the training process, we enable the 'thinking mode' for the Qwen3 model family.

formance gains across various model scales and families. After Stage I, our 1.7B, 7B, and 8B models achieve average performance improvements of 2.40%, 17.40%, and 3.25%, respectively. After Stage II, the IF capabilities are further bolstered, with final average performance increasing by 3.90%, 21.25%, and 5.47%. These consistent improvements validate the effectiveness of PARIF. (2) **IF Performance Gains over Existing Methods.** We conduct comparisons on the 7B model with VerIF and RAIF, and on the 8B model with IFD and LightIF. Results show that our models, following the curriculum learning, exceed these baselines in average performance, and also surpass instruction-tuned models of the same size. Furthermore, our models achieve SOTA results across a wide range of individual tasks. (3) **Strong IF Generalization Performance.** To evaluate the generalizability of our framework, we select two benchmarks that are entirely out-of-distribution relative to our training data: CFBench and Multi-IF. Experimental results demonstrate that our model consistently outperforms the base model on both benchmarks. We attribute these gains to the optimization of the model's reasoning paradigm regarding constraints and the inherent generalizability offered by the RL training paradigm (Kirk et al., 2023). (4) **Pushing the Pareto Frontier.** As shown in Table 10, Stage I leads to a decline in complex reasoning due to its exclusive focus on general-purpose questions with hard constraints. Stage II recovers this performance by incorporating mathematical questions and optimizing logical consistency, ultimately surpassing the baseline methods in overall average performance. Notably, our framework yields consistent gains in coding, a task that demands both deep reasoning and rigid adherence to formal syntax. Figure 4 illustrates the Pareto frontier between general reasoning and IF capabilities across different models. Our framework effectively pushes this Pareto frontier, achieving a superior trade-off between the two. We further verify the statistical robustness of this frontier in Appendix A.5.

4.3 Ablation Studies

Ablation on Stage I. To verify the foundational role of Stage I, we conduct an ablation study by initiating Stage II training directly from the Qwen3-8B model. As shown in Table 2, compared to the complete curriculum learning, bypassing Stage I leads to a significant performance degradation in IF tasks. This provides strong evidence that the optimization of logical consistency in Stage II should be built upon the optimized "reasoning paradigm" established during Stage I.

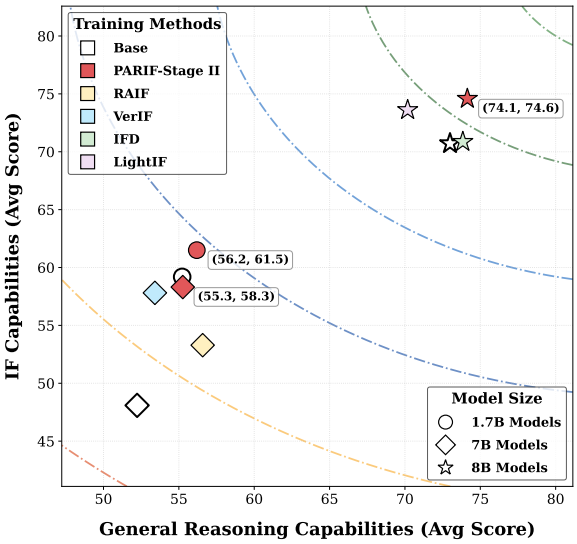


Figure 4: Pareto frontier between IF capabilities and general reasoning capabilities across different models.

Method	FollowB.	IFB.	AIME24	LCB.	Avg.
	SSR	Pr.(S)	A@16	P@1	
PARIF-Stage I	67.11	35.03	68.43	55.78	56.59
PARIF-Stage II	**69.16**	**39.12**	74.50	**56.81**	**59.91**
Directly-Stage II	64.98	33.33	**75.63**	53.61	56.89

Table 2: Comparison of training with and without Stage I on Qwen3-8B. FollowB.: FollowBench; IFB.: IFBench; LCB.: LiveCodeBench v5; A@16: Avg@16; P@1: Pass@1. Best results in each column are **bolded**.

Model	FollowB.	IFB.	AIME24	LCB.	Avg.
	SSR	Pr.(S)	A@16	P@1	
Qwen3-8B	65.22	31.01	74.32	51.38	55.48
PARIF					
- Qwen3-32B	69.16	39.12	74.50	56.81	59.91
- Qwen2.5-14B-Inst.	67.88	37.07	72.50	55.68	58.28
- Llama-3-70B-Inst.	68.89	39.46	73.33	56.25	59.48

Table 3: Comparison results across different verifiers on Qwen3-8B. Abbreviations follow Table 2.

Ablation on Verifier Models. We employ Qwen3-32B as our default verifier model to check the generated answer segments in terms of linguistic fluency, logical correctness, and adherence to soft constraints. We refer to our framework as teacher-free, as we do not directly distill knowledge or capabilities from the verifier model but instead use it solely to provide judgment signals.

To further investigate the framework's sensitivity, we conducted an ablation study using different verifiers on Qwen3-8B model. We select Qwen3-32B, Qwen2.5-14B-Instruct (Qwen et al., 2025), and Meta-Llama-3-70B-Instruct (Grattafiori et al., 2024), spanning varying scales, diverse architectures, and both reasoning and non-reasoning capabilities. As shown in Table 3, our framework achieves consistent performance gains across all tested verifiers. This demonstrates that our approach is agnostic to the specific model family of the verifier and does not rely on its explicit reasoning capabilities.

Ablation on Consistency Score. To validate the effectiveness of the Decoupled-GRPO algorithm, we conducted an ablation study building upon Stage I, comparing our approach with traditional GRPO algorithm based solely on outcome rewards. Specifically, we omit the consistency score and directly employ the rewards derived from Eq. 8 as the outcome rewards to update the policy model. As shown in Table 4, our method outperforms the direct outcome-reward approach across most tasks and achieves a higher average score. This confirms that optimizing logical consistency allows the model to better leverage its reasoning process, thereby yielding gains in both instruction-following and general reasoning tasks.

Method	FollowB.	IFB.	AIME24	LCB.	Avg.
	SSR	Pr.(S)	A@16	P@1	
Qwen3-8B	65.22	31.01	74.32	51.38	55.48
Outcome GRPO	67.87	**40.48**	71.25	54.76	56.36
Decoupled GRPO	**69.16**	39.12	**74.50**	**56.81**	**59.91**

Table 4: Comparison between Outcome-based and Decoupled GRPO. Abbreviations follow Table 2. Best results in each column are **bolded**.

4.4 Analysis

Analysis on Generative Diversity. To verify that generative diversity is preserved alongside improvements in logical consistency, we conduct experiments on two high-difficulty datasets: IFBench and AIME25, representing IF and general reasoning tasks, respectively. We analyze the scaling trends of model performance as the number of samples k increases, as shown in Figure 5. Results indicate that performance on both tasks improves steadily with k. Specifically, our model consistently outperforms the baseline model on IFBench, while on AIME25, it surpasses the baseline model once k exceeds 8. These results demonstrate strong Best-of-N scaling performance. We attribute this success to the inclusion of entropy regularization and KL divergence, which allow the model to maintain an adequate exploration space during reasoning while effectively mitigating distributional collapse.

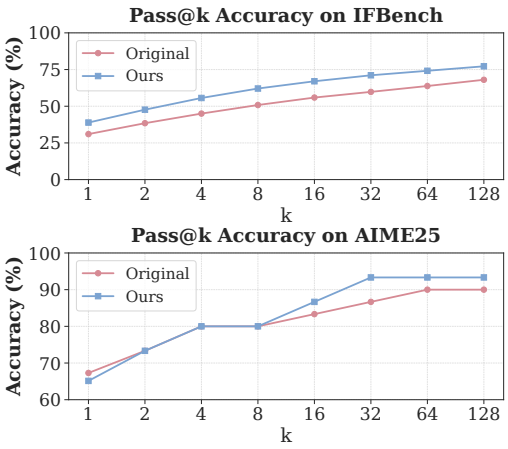


Figure 5: Pass@k results of our Qwen3-8B model.

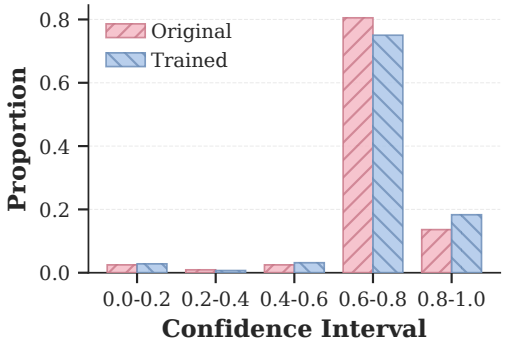


Figure 6: Comparison of confidence distributions for Qwen3-8B model before and after curriculum training on FollowBench, with the y-axis representing the proportion of total samples.

Analysis on Overconfidence Phenomenon. To analyze the impact of consistency scores on model overconfidence, we prompt the model to provide a confidence score at the end of its responses (Yoon et al., 2025), as detailed in Table 20. We compare Qwen3-8B's confidence distribution before and after training (Figure 6). The trained model maintains a healthy distribution; although samples in the extremely high confidence interval (0.8–1.0) increase marginally, the overall volume remains low, while the proportion in lower intervals has grown. We attribute this to external output rewards (Eq. 8), which prevent consistency score inflation,

thus mitigating overconfidence.

5 Conclusion

In this paper, we introduce PARIF, a two-stage curriculum framework that pushes the Pareto frontier of IF and general reasoning for LRMs. Empirical results show that strengthening the logical consistency between reasoning and answers allows models to more effectively utilize their reasoning potential, yielding consistent improvements across multiple benchmarks. We hope that PARIF serves as a robust foundation for future IF research in the era of increasingly powerful reasoning models.

Limitations

Despite PARIF's success in advancing the Pareto frontier of Instruction Following and general reasoning, we acknowledge several limitations. First, our training pipeline primarily targets single-turn instructions. While PARIF demonstrates strong generalization on Multi-IF benchmarks, extending RL benefits to multi-turn dialogues remains a future direction. Second, to mitigate reward hacking, we rely on a computationally heavy Qwen3-32B verifier; future work could explore lightweight, specialized verifiers to improve efficiency. Third, experiments are limited to the 1.7B-8B parameter range due to compute constraints. Finally, the potential of enhancing logical consistency in broader agentic tasks, such as tool-calling and planning, warrants further investigation.

Ethical Considerations

We discuss potential ethical concerns as follows: (1) Intellectual Property and Privacy. We construct our dataset from open-source repositories (ShareGPT, Orca-Chat, LMSYS-Chat, Alpaca, WildChat, and DeepScaleR) (shareAI, 2023; Es, 2023; Zheng et al., 2023; Taori et al., 2023; Zhao et al., 2024; Luo et al., 2025), adhering to their licenses (e.g., Apache 2.0, CC BY-NC 4.0, ODC-By). We filter instructions based on data types (e.g., excluding multi-turn dialogues) and rely on the anonymization protocols of the original datasets to minimize privacy risks. Given the non-commercial nature of some source data, our synthesized dataset will be released under the CC BY-NC 4.0 license. (2) Potential Risk Control. PARIF inherits the known risks of base LLMs (e.g., bias) but focuses on formatting and logical constraints rather than bypassing safety alignments. We do not introduce additional risks. (3) AI Assistance. We utilize AI models to assist in generating verification functions and refining the manuscript. All content is verified by the authors.

Acknowledgments

The research in this article is supported by the National Science Foundation of China (U22B2059, 62276083), Key Research and Development Program of Heilongjiang Providence (2022ZX01A28).

References

Kaikai An, Li Sheng, Ganqu Cui, Shuzheng Si, Ning Ding, Yu Cheng, and Baobao Chang. 2025. Ultraif: Advancing instruction following from the wild. *arXiv preprint arXiv:2502.04153*.

Anthropic. 2025. Claude 3.7 sonnet and claude code. Accessed: 2025-05-06.

Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. 2019. Rudder: Return decomposition for delayed rewards. *Advances in Neural Information Processing Systems*, 32.

Yapei Chang, Yekyung Kim, Michael Krumdick, Amir Zadeh, Chuan Li, Chris Tanner, and Mohit Iyyer. 2025. Bleuberi: Bleu is a surprisingly effective reward for instruction following. *arXiv preprint arXiv:2505.11080*.

Jiale Cheng, Xiao Liu, Cunxiang Wang, Xiaotao Gu, Yida Lu, Dan Zhang, Yuxiao Dong, Jie Tang, Hongning Wang, and Minlie Huang. 2024. Spar: Self-play with tree-search refinement to improve instruction-following in large language models. *arXiv preprint arXiv:2412.11605*.

DeepSeek-AI. 2024. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Guanting Dong, Keming Lu, Chengpeng Li, Tingyu Xia, Bowen Yu, Chang Zhou, and Jingren Zhou. 2024. Self-play with execution feedback: Improving instruction-following capabilities of large language models. *arXiv preprint arXiv:2406.13542*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Ben Elder, Evelyn Duesterwald, and Vinod Muthusamy. 2025. Boosting instruction following at scale. *arXiv preprint arXiv:2510.14842*.

Shahul Es. 2023. Orca-chat: A high-quality explanation-style chat dataset.

Tingchen Fu, Jiawei Gu, Yafu Li, Xiaoye Qu, and Yu Cheng. 2025. Scaling reasoning, losing control: Evaluating instruction following in large reasoning models. *arXiv preprint arXiv:2505.14810*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025a. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Xu Guo, Tianyi Liang, Tong Jian, Xiaogui Yang, Ling-I Wu, Chenhui Li, Zhihui Lu, Qipeng Guo, and Kai Chen. 2025b. Ifdecorator: Wrapping instruction following reinforcement learning with verifiable rewards. *arXiv preprint arXiv:2508.04632*.

Qianyu He, Jie Zeng, Qianxi He, Jiaqing Liang, and Yanghua Xiao. 2024a. From complex to simple: Enhancing multi-constraint complex instruction following ability of large language models. *arXiv preprint arXiv:2404.15846*.

Yun He, Di Jin, Chaoqi Wang, Chloe Bi, Karishma Mandyam, Hejia Zhang, Chen Zhu, Ning Li, Tengyu Xu, Hongjiang Lv, and 1 others. 2024b. Multi-if: Benchmarking llms on multi-turn and multilingual instructions following. *arXiv preprint arXiv:2410.15553*.

Yun He, Wenzhe Li, Hejia Zhang, Songlin Li, Karishma Mandyam, Sopan Khosla, Yuanhao Xiong, Nanshu Wang, Selina Peng, Beibin Li, and 1 others. 2025a. Rubric-based benchmarking and reinforcement learning for advancing llm instruction following. *arXiv preprint arXiv:2511.10507*.

Zirui He, Haiyan Zhao, Yiran Qiao, Fan Yang, Ali Payani, Jing Ma, and Mengnan Du. 2025b. Saif: A sparse autoencoder framework for interpreting and steering instruction following of language models. *arXiv preprint arXiv:2502.11356*.

Juyeon Heo, Christina Heinze-Deml, Oussama Elachqar, Kwan Ho Ryan Chan, Shirley Ren, Udhay Nallasamy, Andy Miller, and Jaya Narain. 2024. Do llms "know" internally when they follow instructions? *arXiv preprint arXiv:2410.14516*.

Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech Czarnecki, Simon Schmitt, and Hado Van Hasselt. 2019. Multi-task deep reinforcement learning with popart. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3796–3803.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.

Daniel Jaroslawicz, Brendan Whiting, Parth Shah, and Karime Maamari. 2025. How many instructions can llms follow at once? *arXiv preprint arXiv:2507.11538*.

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. 2024. Followbench: A multi-level fine-grained constraints following benchmark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4667–4688.

Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. 2023. Understanding the effects of rlhf on llm generalisation and diversity. *arXiv preprint arXiv:2310.06452*.

Xiaoyu Kong, Junguang Jiang, Bin Liu, Ziru Xu, Han Zhu, Jian Xu, Bo Zheng, Jiancan Wu, and Xiang Wang. 2025. Think before recommendation: Autonomous reasoning-enhanced recommender. *arXiv preprint arXiv:2510.23077*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

Hynek Kydlíček. 2025. Math-verify: A library for verifying mathematical answers. `https://github.com/huggingface/Math-Verify`.

Nathan Lambert, Hamish Ivison, Daeyoung Jung, Sage Oore, Alisa Liu, Jacob Morrison, Nouha Dziri, Ari Holtzman, Noah A. Smith, Hannaneh Hajishirzi, and 1 others. 2024a. Tulu 3: Pushing frontiers in open post-training. *arXiv preprint arXiv:2411.15124*.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, and 1 others. 2024b. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*.

Xiaomin Li, Zhou Yu, Zhiwei Zhang, Xupeng Chen, Ziji Zhang, Yingying Zhuang, Narayanan Sadagopan,

and Anurag Beniwal. 2025. When thinking fails: The pitfalls of reasoning for instruction-following in llms. *arXiv preprint arXiv:2505.11423*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024a. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Yilun Liu, Shimin Tao, Xiaofeng Zhao, Ming Zhu, Wenbing Ma, Junhao Zhu, Chang Su, Yutai Hou, Miao Zhang, Min Zhang, and 1 others. 2024b. Coachlm: Automatic instruction revisions improve the data quality in llm instruction tuning. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 5184–5197. IEEE.

Renze Lou, Kai Zhang, and Wenpeng Yin. 2024. Large language model instruction following: A survey of progresses and challenges. *Computational Linguistics*, 50(3):1053–1095.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Li Erran Li, and 1 others. 2025. Deepscaler: Surpassing o1-preview with a 1.5 b model by scaling rl. *Notion Blog*.

math ai. 2025. Aime 2025 dataset.

Maxwell-Jia. 2024. Aime 2024 dataset.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Hao Peng, Yunjia Qi, Xiaozhi Wang, Bin Xu, Lei Hou, and Juanzi Li. 2025. Verif: Verification engineering for reinforcement learning in instruction following. *arXiv preprint arXiv:2506.09942*.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. 2025. Generalizing verifiable instruction following. *arXiv preprint arXiv:2507.02833*.

Yunjia Qi, Hao Peng, Xiaozhi Wang, Bin Xu, Lei Hou, and Juanzi Li. 2025. Constraint back-translation improves complex instruction following of large language models. *Preprint*, arXiv:2410.24175.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*.

Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. 2024a. Infobench: Evaluating instruction following ability in large language models. *arXiv preprint arXiv:2401.03601*.

Yulei Qin, Gang Li, Zongyi Li, Zihan Xu, Yuchen Shi, Zhekai Lin, Xiao Cui, Ke Li, and Xing Sun. 2025a. Incentivizing reasoning for advanced instruction-following of large language models. *arXiv preprint arXiv:2506.01413*.

Yulei Qin, Gang Li, Zongyi Li, Zihan Xu, Yuchen Shi, Zhekai Lin, Xiao Cui, Ke Li, and Xing Sun. 2025b. Incentivizing reasoning for advanced instruction-following of large language models. *Preprint*, arXiv:2506.01413.

Yulei Qin, Yuncheng Yang, Pengcheng Guo, Gang Li, Hang Shao, Yuchen Shi, Zihan Xu, Yun Gu, Ke Li, and Xing Sun. 2024b. Unleashing the power of data tsunami: A comprehensive survey on data assessment and selection for instruction tuning of language models. *Preprint*, arXiv:2408.02085.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Qwen Team. 2025. Qwen3-coder technical report. *arXiv preprint arXiv:2506.05230*.

Qingyu Ren, Qianyu He, Bowei Zhang, Jie Zeng, Jiaqing Liang, Yanghua Xiao, Weikang Zhou, Zeye Sun, and Fei Yu. 2025. Instructions are all you need: Self-supervised reinforcement learning for instruction following. *arXiv preprint arXiv:2510.14420*.

ByteDance Seed, Jiaze Chen, Tiantian Fan, Xin Liu, Lingjun Liu, Zhiqi Lin, Mingxuan Wang, Chengyi Wang, Xiangpeng Wei, Wenyuan Xu, and 1 others. 2025. Seed1. 5-thinking: Advancing superb reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.13914*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

shareAI. 2023. Sharegpt-chinese-english-90k bilingual human-machine qa dataset. https://huggingface.co/datasets/shareAI/ShareGPT-Chinese-English-90k.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Zhan, Yachu Gao, Ding Xi, Shihan Dou, Yifan Wang, Rui Zheng, Enyu Tao, and 1 others. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv:2409.19256*.

Haoran Sun, Lixin Liu, Junjie Li, Fengyu Wang, Baohua Dong, Ran Lin, and Ruohui Huang. 2024. Conifer: Improving complex constrained instruction-following ability of large language models. *Preprint*, arXiv:2404.02823.

Richard Stuart Sutton. 1984. *Temporal credit assignment in reinforcement learning*. University of Massachusetts Amherst.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Doubao Team. 2025. https://team.doubao.com/en/special/doubao_1_5_pro.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

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

Praveen Venkateswaran and Danish Contractor. 2025. Spotlight your instructions: Instruction-following with dynamic attention steering. *arXiv preprint arXiv:2505.12025*.

Vijay Viswanathan, Yanchao Sun, Shuang Ma, Xiang Kong, Meng Cao, Graham Neubig, and Tongshuang Wu. 2025. Checklists are better than reward models for aligning language models. *Preprint*, arXiv:2507.18624.

Chenyang Wang, Liang Wen, Shousheng Jia, Xiangzheng Zhang, and Liang Xu. 2025. Light-if: Endowing llms with generalizable reasoning via preview and self-checking for complex instruction following. *arXiv preprint arXiv:2508.03178*.

Junqiao Wang, Zeng Zhang, Yangfan He, Zihao Zhang, Xinyuan Song, Yuyang Song, Tianyu Shi, Yuchen Li, Hengyuan Xu, Kunyu Wu, and 1 others. 2024a. Enhancing code llms with reinforcement learning in code generation: A survey. *arXiv preprint arXiv:2412.20367*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, and 1 others. 2024b. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290.

Yuning Wu, Jiahao Mei, Ming Yan, Chenliang Li, Shaopeng Lai, Yuran Ren, Zijia Wang, Ji Zhang, Mengyue Wu, Qin Jin, and 1 others. 2025. Writingbench: A comprehensive benchmark for generative writing. *arXiv preprint arXiv:2503.05244*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Dongkeun Yoon, Seungone Kim, Sohee Yang, Sunkyoung Kim, Soyeon Kim, Yongil Kim, Eunbi Choi, Yireun Kim, and Minjoon Seo. 2025. Reasoning models better express their confidence. *arXiv preprint arXiv:2505.14489*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Tao Zhang, Chenglin Zhu, Yanjun Shen, Wenjing Luo, Yan Zhang, Hao Liang, Fan Yang, Mingan Lin, Yujing Qiao, Weipeng Chen, and 1 others. 2025. Cfbench: A comprehensive constraints-following benchmark for llms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32926–32944.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. 2024. Wildchat: 1m chatgpt interaction logs in the wild. *arXiv preprint arXiv:2405.01470*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric P Xing, and 1 others. 2023. Lmsys-chat-1m: A large-scale real-world llm conversation dataset. *arXiv preprint arXiv:2309.11998*.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

A Appendix

A.1 Related Work

Instruction Following Improvements. With the widespread application of Large Language Models, the ability to follow complex instructions has emerged as one of the most critical fundamental capabilities (Lou et al., 2024; Ouyang et al., 2022). One line of research focuses on guiding models by manipulating attention mechanisms or other internal representations (Venkateswaran and Contractor, 2025; Heo et al., 2024; He et al., 2025b). While these approaches offer advantages in terms of cost and efficiency, they do not fundamentally enhance the model's intrinsic understanding of complex constraints. Another prominent direction utilizes

Supervised Fine-Tuning (SFT) or Direct Preference Optimization (DPO) to train models by constructing datasets or data pairs embedded with constraints. The core of these methods lies in data construction, primarily involving techniques such as knowledge distillation (He et al., 2024a; Sun et al., 2024; Dubey et al., 2024; Dong et al., 2024), training specialized models (An et al., 2025; Liu et al., 2024b), and self-play (Cheng et al., 2024). However, acquiring high-quality reference answers and constructing effective positive-negative pairs remain a significant challenge (Jaroslawicz et al., 2025; Elder et al., 2025; Qin et al., 2024b), which ultimately limits the improvement of instruction-following capabilities and hinders generalization. In contrast, our framework eliminates the reliance on high-quality reference responses, while achieving a balance between enhancing instruction following and maintaining general reasoning capabilities.

Reinforcement Learning for Instruction Following Improvement. RLVR has been proven a promising post-training approach (Seed et al., 2025; Wang et al., 2024a; Qian et al., 2025; Kong et al., 2025), with several studies conducting meaningful explorations in the field of instruction following. Specifically, Lambert et al. (2024b); Pyatkin et al. (2025); Wang et al. (2025) leverage the verifiability of hard constraints to perform reinforcement learning. Building upon these efforts, Chang et al. (2025) and Qin et al. (2025a) incorporate soft constraints into the training process; however, they remain dependent on high-quality reference answers, which limits their performance gains. While Ren et al. (2025) train a self-supervised reward model, it lacks an effective mechanism to prevent reward hacking. Although Guo et al. (2025b) and Peng et al. (2025) use external general-purpose models for supervision to mitigate reward hacking, they do not further explore how to restore general reasoning capabilities when applied to reasoning models.

We introduce a curriculum learning method aiming to co-optimize the instruction-following and general reasoning abilities of LRMs. Our framework integrates soft constraints and facilitates teacher-free self-exploration via reinforcement learning. To mitigate reward hacking, we employ a correctness proxy and dynamic constraint weighting, effectively preventing over-optimization. The curriculum is structured to foster deep reasoning regarding constraints in the initial stage, while leveraging the generative probabilities of LRMs to reinforce the logical consistency between the reasoning process and the final answer in the subsequent stage. This approach ensures precise instruction following while preserving the integrity of the model's core reasoning capabilities.

A.2 Dataset Details

Scope of Instruction Following. At the data formulation level, this study involves three core components: seed questions (original queries without constraints, e.g., "Explain today's weather"), constraints (additional formatting or structural requirements for the model's response, e.g., "do not include uppercase letters"), and final instructions (the combined prompt simulating real-world user scenarios). We define this formally as:

$$\text{Final Instruction} = \text{Seed Question} + \text{Constraint(s)}.$$

The scope of instruction following in PARIF requires the model to strictly satisfy the various constraints injected into the final instruction while correctly resolving the core intent of the seed question itself. This aligns directly with our RLVR training objectives.

Data Construction Details.

- **Seed Collection:** We employ a heuristic filtering strategy to refine the seed dataset. Specifically, we exclude multi-turn dialogues, excessively long questions, non-English content, purely instructional questions, programming or code-related questions, and mathematical questions. The specific prompts utilized for this filtering process are detailed in Table 15.

- **Constraint Construction:** We categorize our collected hard constraint templates into six categories: Length (constraining response length), Keyword (requiring specific terms), Lexical (lexical-level requirements), Language (specifying the output language), Format (structural requirements), and Affix (dictating start or end sequences). For each template, we utilize Qwen3-Coder to synthesize corresponding verification functions, treating template placeholders as function arguments. Regarding soft constraints, we follow the methodology of VerIF (Peng et al., 2025) by first prompting an LLM (Qwen3-235B-A22B-Instruct-2507) to generate a response to the original question, and then extracting implicit soft constraints satisfied by that response.

- **Data Generation:** We enumerate combinations of hard constraint templates via a Cartesian product. To mitigate combinatorial explosion and preclude logical contradictions, we limit each set to a maximum of six templates from distinct categories. We perform conflict detection between the injected constraints and the original question to ensure logical consistency. The prompt is provided in Table 16. Given the inherent difficulty LLMs face with precise length adherence (Sun et al., 2024; Zhou et al., 2023), we initially omit the length constraint templates, generate a preliminary response, and then backfill the length constraint templates based on the actual output length. In contrast, other constraint templates are dynamically populated directly based on the original question. The prompt is provided in Table 17. To eliminate positional bias, we randomize the placement of the questions and constraints and derive the final instructions via string concatenation rather than LLM rewriting to avoid unintended semantic drift. Post-concatenation, we use an LLM judge to filter for semantic coherence. In Stage II, we incorporate mathematical questions. To avoid excessive reasoning overhead and logical conflicts, we select a specialized subset of hard constraints suitable for mathematical tasks (Table 18) and restrict each question to a single additional constraint.

Dataset Statistics. We provide the statistical distribution of various constraints and the proportion of instructions categorized by their constraint counts within the dataset. Figure 7 illustrates the statistics of our dataset.

Extension of Verifiable Constraint Categories. Compared to other works that leverage RL to enhance models' instruction following capabilities, our taxonomy significantly broadens the verifiable scope:

- **Compared to AutoIF (Dong et al., 2024):** We extend the Python-based verification categories to include Keyword, Language, and Affix constraints. AutoIF primarily focuses on Lexical and Format constraints, with limited exploration of Length.
- **Compared to VerIF (Peng et al., 2025):** We extend Language constraints, Affix constraints, and Lexical constraints.
- **Compared to CheckList (Viswanathan et al., 2025):** We extend the taxonomy to include Length, Language, and Affix constraints.

In summary, our taxonomy extends prior work to cover a much broader and more scalable range of practical user scenarios. Nevertheless, we acknowledge that defining a universal constraint taxonomy remains an evolving challenge in the field. We plan to continuously expand and refine this system in future work to better capture the complexities of real-world interactions.

A.3 Framework Details

Data Allocation. We allocate specialized training data across different stages to enhance diverse model capabilities.

- **Stage I:** In this stage, we primarily focus on bolstering the model's foundational IF capabilities and optimizing its reasoning paradigm relative to user instructions (refer to Appendix A.5 for more details). This phase utilizes 16k general instructions exclusively containing hard constraints. Compared to soft constraints, hard ones offer significantly lower evaluation noise during the verification process. Furthermore, since hard constraints prioritize rule execution rather than relying on complex external world knowledge or profound semantic nuances, they serve as a more direct and intuitive proxy for reflecting and enhancing the model's core IF capabilities.
- **Stage II:** In this stage, our objectives are to bolster the model's general reasoning capabilities and further generalize its IF capabilities. We incorporate 5k mathematical instructions alongside 5k general instructions with soft constraints. Mathematical questions are stochastically drawn from DeepScaleR (Luo et al., 2025), with only one hard constraint injected into each question to preclude logical conflicts. For general instructions, we ensure that each instruction contains at least one soft constraint, and the total number of constraints is capped at ten. The selected math questions possess explicit ground-truth answers. Compared to general questions, their correctness verification involves less noise, which makes it more difficult for models to deceive the verifier for reward hacking.

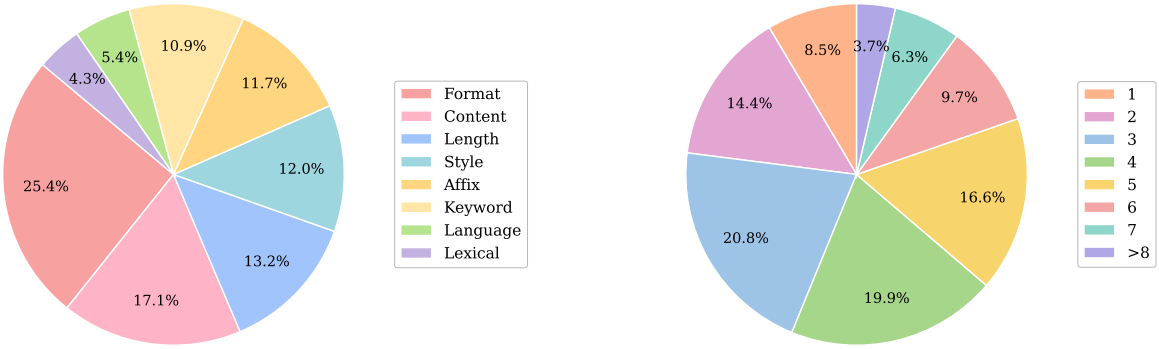


Figure 7: Dataset statistics. **Left:** Distribution of various constraint types. **Right:** Proportion of instructions categorized by the number of constraints.

Reward Design Rationale and Details. In this part, we detail the rationale behind the reward design at different stages of our framework.

- **Stage I:** In this stage, we implement multiple measures against reward hacking (details in Section 3.2). We use a correctness proxy to evaluate both the linguist-level and logic-level correctness of the model's outputs. To minimize computational overhead, we adopt a cascaded verification strategy: we prioritize linguist-level verification, using it as a gate to determine whether a subsequent check for logic-level correctness is necessary. We leverage GPT-4o to analyze model outputs across various IF tasks, synthesizing observed reward hacking patterns to construct a pre-defined checklist to assist the verification. During this check, we use the original question without any constraints. This practice mitigates the risk of the LLM verifier erroneously executing the constraints itself, thereby ensuring better alignment with the core user intent. The corresponding prompt is provided in Table 19. Upon successful validation, the process moves forward to the logic-level correctness check. The prompt is provided in Table 21. Furthermore, we employ dense rewards for constraints in this stage. This provides the model with frequent feedback, ensuring the curriculum learning framework initializes at a manageable difficulty level.

- **Stage II:** To further catalyze the model's IF capabilities, we employ sparse reward signals for constraint adherence in this stage, thereby intensifying the curriculum difficulty. A core contribution of this stage is the promotion of logical consistency between the reasoning process and the answer segment. We introduce a consistency score, which is defined as the posterior probability of the answer segment conditioned on the reasoning process. Averaging the token output probabilities ensures that the consistency score calculation is not affected by sequence length, while constraining the values to the [0, 1] range to align with other reward signals. Traditional outcome-based rewards adopt an extreme assignment strategy, directly attributing the reward from the answer segment to the reasoning process with a fixed coefficient of 1. In contrast, our framework utilizes the consistency score as a dynamic coefficient for reward distribution. Since the binary outcome reward r serves as a supervisory signal, the model is precluded from obtaining higher rewards through the simple inflation of consistency scores. Unlike explicit calibration methods, we do not intentionally suppress consistency; rather, our objective function encourages the model to prioritize optimizing the binary reward r and to further refine logical consistency specifically when $r = 1$.

Consistency Score Design Rationale and Details. In Section 3.3, we compute the consistency score via Eq. 10 as the average probability of the answer tokens. Rather than directly measuring the quality of the reasoning process, this score captures the logical consistency between the reasoning process and the answer segment, and is optimized within our Decoupled-GRPO algorithm to promote their alignment. Mathematically, this corresponds to the posterior probability $P(y \mid x, z)$ of answer y given

problem x and reasoning process z. To validate that this score reflects logical consistency, we conduct additional verification experiments. Specifically, for a given problem, we sample 64 responses from the model and swap the answers across different reasoning processes. As shown in Table 5, the consistency score drops significantly after swapping, confirming that it effectively captures logical consistency. We further observe that the consistency score also measures model certainty in a formal sense. Importantly, certainty and consistency are not conflicting notions — they play distinct yet complementary roles within our framework. The consistency score promotes reasoning-answer alignment through the lens of logical consistency, while certainty provides an interpretation of the experimental phenomena. Concretely, Figure 3 shows that a correct reasoning process typically yields a more certain answer, meaning the consistency score serves as an implicit proxy for reasoning quality from the certainty perspective. This explains why explicitly optimizing the consistency score yields improvements over outcome-reward-based GRPO on reasoning-intensive tasks such as mathematics and coding (Table 4).

Model	AIME25	IFBench
Qwen3-8B	0.92 → 0.75 (-18.48%)	0.91 → 0.80 (-12.09%)
R1-Distill-Qwen-7B	0.95 → 0.78 (-17.89%)	0.86 → 0.73 (-15.12%)

Table 5: Consistency score before and after answer swapping across different reasoning processes. Each cell shows the original probability → swapped probability (percentage decrease).

Decoupled GRPO Algorithm Details. The complete procedure of the Decoupled GRPO is presented in Algorithm 1. We compute $\mathcal{J}_{Reason}$ and $\mathcal{J}_{Answer}$ based on Eq. 4, incorporating a KL divergence regularization term. This KL constraint prevents policy degeneration, specifically prohibiting the model from simply outputting high-probability tokens to exploit the consistency metric without genuine reasoning. Subsequently, we update the policy by Eq. 13, introducing an entropy term into the optimization. A critical design choice is that we calculate this entropy **solely over the reasoning tokens** (reasoning process), rather than the entire response. On one hand, we use this to preserve the model's capability for exploration and diversity during the reasoning process. On the other hand, excluding answer tokens ensures that this entropy maximization does not conflict with the objective

Algorithm 1 Decoupled-GRPO

Input stage I checkpoint π_θ; verifier model R; task instructions $\mathcal{D}$; entropy coefficient α
  for *step* $= 1, \ldots, M$ **do**
    Sample batch $\mathcal{D}_b$ from $\mathcal{D}$ and update reference policy $\pi_{\theta_{\text{old}}} \leftarrow \pi_\theta$
    Generate G outputs $\{o_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot|x)$ for each $x \in \mathcal{D}_b$
    for *each output* o_i **do**
      Get its answer segment y_i
      Compute reward $r(q, y_i)$ by running R (Eq. 8)
      Compute consistency score $\text{Cons}(o_i)$ (Eq. 10)
      Compute decoupled rewards $r^{(i)}_{\text{reason}}, r^{(i)}_{\text{answer}}$
    end
    for *each output* o_i **do**
      Compute $\hat{A}^{(i)}_{reason}$ for each reasoning token of o_i (Eq. 11)
      Compute $\hat{A}^{(i)}_{answer}$ for each answer token of o_i (Eq. 12)
    end
    for *iteration* $= 1, \ldots, \mu$ **do**
      Compute $\mathcal{J}_{\text{Reason}}$ and $\mathcal{J}_{\text{Answer}}$ on the reasoning process and the answer segment, respectively (Eq. 4)
      Update policy π_θ by maximizing $\mathcal{J}_{\text{Total}}$ with α (Eq. 13)
    end
  end
Output : π_θ

of enhancing logical consistency. Experimental results demonstrate that while we improve logical consistency by increasing the output probability of the answer segment, the increase in logical consistency enables the answers to better align with diverse reasoning processes, thereby maintaining the model's Best-of-N (BoN) performance (see Section 4.4).

A.4 Experiment Details

Implementation Details. Our implementation is built upon the verl framework (Sheng et al., 2024), utilizing vLLM (Kwon et al., 2023) as the inference backend. The core training process is based on the Group Relative Policy Optimization (GRPO) algorithm. Throughout both training stages, we employ Group Normalization, an elevated clipping threshold ϵ_{high}, and token-level loss aggregation. To mitigate any potential influence or bias from the system prompt on the model's intrinsic IF capabilities, we intentionally omit the system prompt throughout the experiment. For mathematical questions with explicit numerical ground-truth answers, we use Math-verify (Kydlíček, 2025) to validate

the correctness of model responses. In cases where the length of the reasoning process exceeds the maximum token limit, we set the answer segment to an empty string. Key hyperparameters and configurations are summarized in Table 6.

Hyperparameter	Value
Sampling Configuration	
backend	vLLM
temperature	1.0
max prompt length	2048
max response length (Stage I)	10240
max response length (Stage II)	24576
rollout size	16
gpu_memory_utilization	0.7
LLM verification Configuration	
default model	Qwen3-32B
backend	vLLM
temperature	0.6
max tokens	8192
tensor-parallel-size(vLLM)	2
Training Configuration	
train batch size (Stage I)	160
train batch size (Stage II)	64
learning rate	1.0e-06
lr scheduler type	constant
ϵ_{low}	0.2
ϵ_{high} (Stage I)	0.26
ϵ_{high} (Stage II)	0.28
kl loss coef	1.0e-03
entropy coeff (Stage I)	0
entropy coeff (Stage II)	1.0e-03
Evaluation Configuration	
max tokens	32768
temperature	0.6
top P	0.95
top K	-1
gpu_memory_utilization	1.0

Table 6: Hyperparameter settings and configurations for training and evaluation.

Detailed Results on IF Benchmarks. Tables 7, 8, and 9 present the detailed evaluation results for FollowBench (HSR), FollowBench (SSR), and InFoBench, respectively. After Stage I training, our models achieve consistent performance gains over their corresponding original models. Stage II builds upon these foundations to further enhance results; ultimately, the average performance of our models exceeds that of instruction-tuned models of the same scale and surpasses the vast majority of baseline methods.

Results on General Benchmarks. Table 10 presents the evaluation results on general benchmarks. As Stage I exclusively focuses on general questions with only hard constraints, the models exhibit a performance decline in complex mathematical reasoning tasks, such as AIME24 and AIME25. In Stage II, we introduce mathematical questions and enhance the logical consistency between the reasoning process and the answers, thereby recovering and further improving the models' performance.

Benchmark Details. We selected a total of six IF benchmarks and six general reasoning datasets.

- **IFEval** (Zhou et al., 2023) is designed to evaluate the instruction-following capabilities of LLMs. This benchmark comprises 541 prompts covering 25 types of automatically verifiable instructions. In our evaluation, we focus on the prompt-level metric and employ both loose and strict accuracy scores to measure model performance.

- **FollowBench** (Jiang et al., 2024) is a benchmark designed to evaluate the multi-level fine-grained instruction-following capabilities of LLMs. It incorporates five distinct categories of constraints: Content, Situation, Style, Format, and Example. In our experiments, we employ an LLM-as-a-judge approach and report two primary metrics: Hard Satisfaction Rate (HSR), defined as the average rate at which all constraints within an individual instruction are fully satisfied , and Soft Satisfaction Rate (SSR), which calculates the average satisfaction rate of individual constraints across all instructions.

- **IFBench** (Pyatkin et al., 2025) is a benchmark designed to evaluate the generalization capabilities of LLMs in instruction following with unseen constraints. Addressing the issues of performance saturation and model overfitting observed in existing benchmarks, IFBench introduces 58 novel and challenging out-of-distribution (OOD) verifiable constraints spanning 7 major categories, including Count, Ratio, Words, and Format. In our evaluation, we report both strict and loose accuracy metrics at the prompt level.

- **InFoBench** (Qin et al., 2024a) is a benchmark dataset comprising 500 diverse instructions and 2,250 corresponding decomposed questions, covering various types of instruction constraints. This benchmark introduces a novel evaluation metric termed the Decomposed Requirements Following Ratio (DRFR) to assess the capability of models in following complex instructions. In

Model	FollowBench (HSR)					Avg.
	Level 1	Level 2	Level 3	Level 4	Level 5	
***Large Baseline Models* ($\geq$ 70B)**						
Qwen2.5-72B-Inst.	77.34	68.30	57.04	57.70	52.51	62.58
GPT-4o	78.31	69.74	61.93	60.49	53.75	64.84
DeepSeek-V3	85.43	76.72	71.07	68.14	65.04	73.28
Doubao-1.5-pro	70.53	68.18	66.19	56.58	50.74	62.44
Instruction-Tuned Baseline Models						
Qwen2.5-7B-Inst.	47.52	45.24	30.34	31.76	19.28	34.83
Mistral-Crab-7B-DPO	66.10	53.60	53.40	42.40	31.70	49.44
Llama-3.1-Tulu-3-8B	74.61	69.57	62.97	53.70	49.96	62.16
Llama-3.1-Tulu-3-8B-VerIF	73.36	68.16	55.71	49.19	46.33	58.55
Reasoning Models (1.7B)						
Qwen3-1.7B	67.50	59.17	47.03	47.23	37.12	51.61
PARIF-Stage I	64.91	56.57	50.93	45.93	42.02	52.07
PARIF-Stage II	68.19	61.94	52.43	46.78	41.73	54.21
Reasoning Models (7B)						
R1-Distill-Qwen-7B	56.98	55.24	40.40	37.61	28.87	43.82
R1-Distill-Qwen-7B-RAIF	62.69	54.56	46.44	50.77	44.33	51.76
R1-Distill-Qwen-7B-VerIF	63.34	57.88	50.49	46.21	35.92	50.77
PARIF-Stage I	63.25	56.83	46.46	42.16	30.47	47.83
PARIF-Stage II	63.75	59.99	49.27	42.64	40.34	51.20
Reasoning Models (8B)						
Qwen3-8B	74.88	63.85	59.33	58.75	53.40	62.04
Qwen3-8B-IFD	69.74	65.78	57.98	58.17	50.02	60.34
Qwen3-8B-LightIF	74.61	70.10	63.64	55.85	56.00	64.04
PARIF-Stage I	73.30	68.21	62.80	58.92	53.95	63.44
PARIF-Stage II	78.98	68.68	62.62	63.81	56.20	66.06

Table 7: Detailed evaluation results on FollowBench (HSR).

this study, we employ an LLM-as-a-judge approach. We report results on the Easy Set, Hard Set, and Overall.

- **CFBench** (Zhang et al., 2025) is a large-scale, comprehensive Chinese benchmark designed to evaluate the instruction-following capabilities of LLMs across diverse and realistic scenarios. The benchmark spans 20 real-world domains and over 50 Natural Language Processing tasks, comprising 1,000 meticulously curated samples. In this study, we employ an LLM-as-a-judge approach, utilizing a fine-grained checklist mechanism for evaluation. We report three primary metrics: Constraint Satisfaction Rate (CSR), Instruction Satisfaction Rate (ISR), and Priority Satisfaction Rate (PSR).

- **Multi-IF** (He et al., 2024b) is a benchmark designed to evaluate the instruction-following capabilities of LLMs in multi-turn and multilingual contexts. The benchmark comprises 4,501 conversation samples spanning 8 languages, with each conversation consisting of three turns. In our experiments, we report the comprehensive average accuracy across all languages for each turn (calculated as the average of instruction-level and conversation-level accuracy under both strict and loose criteria).

- **Math500** (Lightman et al., 2023) is a subset of the MATH dataset, consisting of 500 problems that span multiple mathematical disciplines. We utilize Math-Verify to evaluate the correctness of the model's answers and report the overall accuracy.

- **AIME24 & AIME25** (Maxwell-Jia, 2024; math ai, 2025) originate from the American Invitational Mathematics Examination. Each dataset contains 30 Olympiad-level challenging problems covering diverse domains such as algebra and geometry. In this study, we employ Math-Verify for evaluation. To ensure the stability of our results, we report the avg@16 metric.

- **MMLU-Pro** (Wang et al., 2024b) builds upon the Massive Multitask Language Understanding (MMLU) dataset by incorporating more challenging, reasoning-focused questions. It is designed to comprehensively evaluate the capabilities of language models in complex knowledge

Model	FollowBench (SSR)					Avg.
	Level 1	Level 2	Level 3	Level 4	Level 5	
Large Baseline Models ($\geq$ 70B)						
Qwen2.5-72B-Inst.	77.34	71.05	61.86	62.19	64.02	67.29
GPT-4o	78.31	73.06	67.98	65.88	64.01	69.85
DeepSeek-V3	85.43	77.88	73.86	71.11	71.15	75.89
Doubao-1.5-pro	70.53	70.93	70.94	62.80	64.44	67.93
Instruction-Tuned Baseline Models						
Qwen2.5-7B-Inst.	47.52	52.30	42.00	46.12	41.11	45.81
Mistral-Crab-7B-DPO	64.01	61.45	57.29	48.11	51.58	56.49
Llama-3.1-Tulu-3-8B	74.61	72.44	66.92	59.79	61.38	67.03
Llama-3.1-Tulu-3-8B-VerIF	73.36	70.87	61.60	56.08	59.78	64.34
Reasoning Models (1.7B)						
Qwen3-1.7B	67.50	63.88	54.81	54.83	49.88	58.18
PARIF-Stage I	64.91	61.06	58.10	54.46	54.63	58.63
PARIF-Stage II	68.19	64.03	58.13	54.55	54.58	59.90
Reasoning Models (7B)						
R1-Distill-Qwen-7B	56.98	60.70	51.41	50.27	49.04	53.68
R1-Distill-Qwen-7B-RAIF	62.69	60.22	54.14	55.83	56.32	57.84
R1-Distill-Qwen-7B-VerIF	63.34	63.35	58.57	57.12	52.44	58.96
PARIF-Stage I	63.25	62.53	55.14	53.06	52.33	57.26
PARIF-Stage II	63.75	64.43	58.00	54.89	54.02	59.02
Reasoning Models (8B)						
Qwen3-8B	74.88	67.11	61.84	62.07	60.19	65.22
Qwen3-8B-IFD	69.74	68.09	62.24	62.89	56.52	63.90
Qwen3-8B-LightIF	74.61	71.22	66.73	61.91	65.13	67.92
PARIF-Stage I	73.30	69.88	65.81	62.65	63.91	67.11
PARIF-Stage II	78.98	70.61	64.94	66.92	64.33	69.16

Table 8: Detailed evaluation results on FollowBench (SSR).

understanding and reasoning tasks. The benchmark comprises over 12,000 rigorously filtered questions derived from academic examinations and authoritative textbooks, spanning 14 diverse disciplinary domains. In this study, we report the overall accuracy of the models on this benchmark.

- **WritingBench** (Wu et al., 2025) is a comprehensive benchmark designed to evaluate the generative writing capabilities of LLMs. It comprises 1,000 real-world queries spanning 6 primary domains and 100 fine-grained subdomains, where each query is paired with five task-specific scoring criteria. In this study, we employ the official critic model (WritingBench-Critic-Model-Qwen2.5-7B) to conduct automatic evaluation.

- **LiveCodeBench** (v5, 2024.10-2025.02) (Jain et al., 2024) is employed to assess coding proficiency. Consistent with (Yang et al., 2025), the constraint "You will not return anything except for the program" is omitted to facilitate unconstrained reasoning during inference. The pass@1 metric is adopted for evaluation.

A.5 Further Analyses

Shift in Reasoning Patterns. We assess the deep thinking rates of Qwen3-8B and DeepSeek-R1-Distill-Qwen-7B on IFEval and IFBench to detect changes in reasoning paradigms at various training phases. The results and prompts are shown in Table 11 and Table 22, respectively. Results show that Stage I training significantly boosts the model's deep thinking rate, evolving from simple constraint restatement to deeper logical engagement. This performance is preserved or even improved upon completion of Stage II.

Statistical Robustness of the Pareto Frontier. To verify that PARIF's advancement of the Pareto frontier is statistically robust, we conduct a bootstrap-based probabilistic analysis. Specifically, we normalize all benchmark scores relative to a "Random Guess Baseline" (mapped to 0.0) and a "Baseline Model of Corresponding Scale" (mapped to 1.0), using R1-Distill-Qwen-7B as the reference for the 7B group and Qwen3-8B for the 8B group, which eliminates scale discrepancies across heterogeneous datasets. We then perform 1,000 iterations of bootstrap resampling *over the set of*

Model	InFoBench		
	Easy	Hard	Overall
Large Baseline Models ($\geq$ 70B)			
Qwen2.5-72B-Inst.	83.77	83.14	83.33
GPT-4o	82.46	82.63	82.58
DeepSeek-V3	85.80	87.24	86.80
Doubao-1.5-pro	82.69	81.45	82.31
Instruction-Tuned Baseline Models			
Qwen2.5-7B-Inst.	63.48	70.13	68.09
Mistral-Crab-7B-DPO	78.84	77.50	77.91
Llama-3.1-Tulu-3-8B	80.43	82.24	81.69
Llama-3.1-Tulu-3-8B-VerIF	80.14	79.17	79.47
Reasoning Models (1.7B)			
Qwen3-1.7B	80.72	81.86	81.51
PARIF-Stage I	82.61	82.82	82.76
PARIF-Stage II	83.36	82.88	83.03
Reasoning Models (7B)			
R1-Distill-Qwen-7B	75.07	76.03	75.73
R1-Distill-Qwen-7B-RAIF	79.42	79.94	79.78
R1-Distill-Qwen-7B-VerIF	78.55	80.13	79.64
PARIF-Stage I	77.54	80.51	79.60
PARIF-Stage II	77.29	81.19	79.88
Reasoning Models (8B)			
Qwen3-8B	86.09	85.71	85.82
Qwen3-8B-IFD	84.78	87.69	86.80
Qwen3-8B-LightIF	85.51	85.58	85.56
PARIF-Stage I	86.06	84.66	85.13
PARIF-Stage II	87.35	86.36	86.69

Table 9: Detailed evaluation results on InFoBench.

evaluation benchmarks (i.e., in each iteration we draw K benchmarks uniformly with replacement from the K available benchmarks); in each iteration we recompute every model's composite score as the mean of its normalized scores across the resampled benchmarks, and re-determine its Pareto frontier membership. We report the mean and 95% confidence interval (CI) of the resulting composite scores. We further define the **Pareto Stability Score** as the probability (in $[0, 1]$) that a model lies on the Pareto frontier across the $1{,}000$ bootstrap iterations, where a score of 1.0 indicates that the model remains undominated regardless of benchmark selection fluctuations.

As shown in Table 12, PARIF achieves 100% **Pareto Stability** at both the 7B and 8B scales. Although some baselines (e.g., R1-Distill-Qwen-7B-VerIF) exhibit slight advantages on individual metrics with non-trivial variance, only PARIF maintains an undominated frontier position across all bootstrap samples while remaining competitive on both axes. This confirms that PARIF's advancement of the Pareto frontier is statistically robust rather than an accidental result tied to a specific benchmark configuration.

Analysis of Training Dynamics. Figure 8 illustrates the training dynamics of our 7B and 8B models across different stages, encompassing changes in total rewards, response length, and consistency scores. The results indicate that total rewards grow steadily with training steps across stages, demonstrating the framework's stability. In Stage I, the response length decreases as training progresses. We attribute this to our exclusive use of general-purpose training data, which directs the model toward precise instruction following with explicit constraints rather than the divergent thinking typical of mathematical reasoning. In Stage II, with the introduction of mathematical data, the response length no longer exhibits a monotonic decline. Influenced by the first-stage training, the model achieves a balance between exploratory divergent thinking and structured adherence, eventually stabilizing response length within a reasonable range. Although we enhance the logical consistency between the reasoning process and the answer seg-

Model	MATH-500	AIME24	AIME25	MMLU-Pro	WritingBench	LiveCodeBench v5	Avg.
	Pass@1	Avg@16	Avg@16	Pass@1	Overall	Pass@1	
Reasoning Models (1.7B)							
Qwen3-1.7B	91.60	50.00	35.83	55.19	6.98	28.91	55.22
PARIF-Stage I	91.00	48.54	34.79	55.03	7.05	28.55	54.74 (-0.87%)
PARIF-Stage II	**92.20**	**50.42**	**37.50**	**55.39**	**7.21**	**29.52**	**56.19** (+1.76%)
Reasoning Models (7B)							
R1-Distill-Qwen-7B	93.20	54.38	37.92	48.24	5.23	27.34	52.23
R1-Distill-Qwen-7B-RAIF	93.20	**56.25**	39.38	**54.90**	**6.44**	31.33	**56.58** (+8.32%)
R1-Distill-Qwen-7B-VerIF	93.60	55.60	40.42	46.72	5.48	29.33	53.41 (+2.26%)
PARIF-Stage I	93.40	49.16	35.20	51.18	5.58	31.93	52.78 (+1.05%)
PARIF-Stage II	**94.60**	53.96	**41.67**	51.26	5.80	**32.06**	55.26 (+5.80%)
Reasoning Models (8B)							
Qwen3-8B	96.80	74.32	**67.29**	73.15	7.50	51.38	72.99
Qwen3-8B-IFD	**97.20**	74.17	65.00	**74.47**	7.73	54.87	73.84 (+1.16%)
Qwen3-8B-LightIF	95.40	68.50	63.75	73.73	6.80	51.69	70.18 (-3.85%)
PARIF-Stage I	96.00	68.43	60.10	73.41	7.61	55.78	71.64 (-1.85%)
PARIF-Stage II	**97.20**	**74.50**	65.13	73.20	**7.80**	**56.81**	**74.14** (+1.58%)

Table 10: Main results (%) on general benchmarks. Due to the limited sample size of AIME24 and AIME25 (30 problems each), we report the avg@16 results to ensure a more stable evaluation of model capabilities. We remove the reasoning process and evaluate using only the answer segment. Note that while WritingBench is reported on a 10-point scale, it is multiplied by 10 for the average calculation to align with the percentage scale. Best/2nd best are **bolded**/underlined.

Method	IFEval	IFBench
Qwen3-8B	87.78	87.71
Qwen3-8B-Stage I	92.78	90.48
Qwen3-8B-Stage II	94.63	91.84
DS-7B	73.53	63.01
DS-7B-Stage I	86.64	77.21
DS-7B-Stage II	87.20	77.13

Table 11: Evolution of deep thinking rates across different training stages. DS denotes DeepSeek-R1-Distill-Qwen.

ment in Stage II, the consistency score does not suffer from rapid overfitting. This is attributed to three primary factors. (1) Outcome Rewards. The presence of outcome rewards prevents the model from gaining higher rewards solely by inflating the consistency score. (2) Entropy Regularization. The entropy regularization term preserves exploratory capacity, discouraging the model from "cheating" for rewards by simply outputting high-probability tokens. (3) KL Divergence. The inclusion of KL divergence protects the model's output distribution from degradation.

Comparison with Supervised Fine-Tuning (SFT). To investigate whether our proposed framework offers advantages over the traditionally dominant SFT-based approach, we leverage Qwen3-235B-A22B-Thinking-2507 to synthesize responses for our dataset, including both reasoning processes and final answers. We then fine-tune the Qwen3-8B model using this data, with the results presented in Table 13. The results indicate that, compared to our approach, the supervised fine-tuned model demonstrates limited improvements on in-distribution IF benchmarks and suffers from performance degradation on out-of-distribution IF and general reasoning benchmarks, particularly in complex reasoning tasks. This reveals the insufficient generalization of traditional SFT methods and their failure to strike an effective balance between IF capabilities and general reasoning abilities. In contrast, our framework does not rely on teacher-model answer sampling, exhibits robust generalization, and achieves superior performance gains even when utilizing a relatively smaller external model (32B vs. 235B).

Data Contamination Analysis. To ensure that the performance gains are not derived from training data leakage or rote memorization of test instances, we conduct a comprehensive contamination analysis of our Stage I and Stage II training data. This analysis is performed across six benchmarks covering both instruction following (IFEval, IFBench, FollowBench) and general reasoning (AIME24, AIME25, WritingBench). Specifically, we employ two complementary methodologies:

- **N-gram Overlap:** We utilize an N-gram overlap method with $N = 13$ to identify exact matches or highly similar textual sequences between the training and test data.

Model	Scale	IF Norm. Mean	Reasoning Norm. Mean	Pareto Stability
R1-Distill-Qwen-7B (Base)	7B	1.0000 [1.0000, 1.0000]	1.0000 [1.0000, 1.0000]	0.000 (0.0%)
R1-Distill-Qwen-7B-RAIF	7B	1.1469 [1.0965, 1.2108]	1.0989 [1.0402, 1.1664]	0.810 (81.0%)
R1-Distill-Qwen-7B-VerIF	7B	1.3063 [1.1720, 1.5015]	1.0301 [1.0009, 1.0568]	0.016 (1.6%)
PARIF-7B-Stage II (Ours)	7B	**1.3264 [1.1838, 1.5393]**	**1.0754 [1.0293, 1.1208]**	**1.000 (100.0%)**
Qwen3-8B (Base)	8B	1.0000 [1.0000, 1.0000]	1.0000 [1.0000, 1.0000]	0.107 (10.7%)
Qwen3-8B-IFD	8B	0.9911 [0.9584, 1.0163]	1.0144 [0.9908, 1.0383]	0.147 (14.7%)
Qwen3-8B-LightIF	8B	1.0595 [1.0171, 1.1147]	0.9627 [0.9316, 0.9929]	0.007 (0.7%)
PARIF-8B-Stage II (Ours)	8B	**1.0726 [1.0324, 1.1215]**	**1.0204 [0.9911, 1.0546]**	**1.000 (100.0%)**

Table 12: Bootstrap-based statistical robustness analysis of the Pareto frontier. We perform $N = 1{,}000$ bootstrap iterations by resampling the set of evaluation benchmarks with replacement; results are reported as Mean $[95\%$ CI$]$ of the resulting composite scores. Scores are normalized against a Random Guess Baseline (0.0) and the corresponding base model (1.0). **Pareto Stability** denotes the probability that a model remains on the Pareto frontier across the $1{,}000$ iterations.

Method	IFEval	FollowB.	IFB.	InfoB.	CFB.	Multi.	AIME24	AIME25	MMLU-P.	LCB.	Avg.
	Pr.(S)	SSR	Pr.(S)	Overall	PSR	step3	A@16	A@16	P@1	P@1	
Qwen3-8B	83.73	65.22	31.01	85.82	**75**	71.76	74.32	**67.29**	73.15	51.38	67.87
Qwen3-8B-SFT	85.36	68.19	37.07	84.71	74	68.52	53.96	52.92	72.47	46.63	64.38
Qwen3-8B-PARIF	**89.78**	**69.16**	**39.12**	**86.69**	**75**	**76.43**	**74.50**	65.13	**73.20**	**56.81**	**70.61**

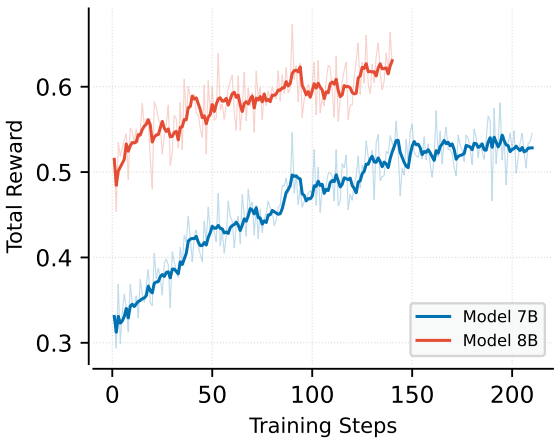
Table 13: Comparative results of Qwen3-8B across various tasks after fine-tuning and PARIF curriculum learning. Evaluation covers both IF and general benchmarks. InfoB.: InfoBench; CFB.: CFBench; Multi.: Multi-IF; MMLU-P.: MMLU-Pro. Other abbreviations follow Table 2. Best results in each column are **bolded**.

- **Semantic Similarity Analysis:** To capture paraphrased or semantically rewritten samples that might evade N-gram detection, we leverage Sentence Transformers (all-MiniLM-L6-v2) to compute text embeddings. We establish a cosine similarity threshold of $\tau = 0.9$; samples exceeding this value are considered potentially contaminated.

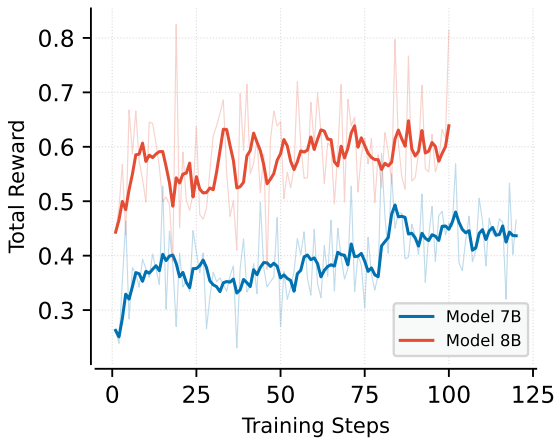
As illustrated in Table 14, the results confirm that contamination levels across all benchmarks are minimal. Specifically, Stage I data exhibits marginal overlap in instruction-following benchmarks (e.g., 1.68% N-gram overlap on IFEval), consistent with its training objective. Correspondingly, Stage II data shows slight overlaps in mathematical reasoning benchmarks (e.g., 2.52% on AIME24). Notably, for these reasoning tasks, the semantic similarity ratios are significantly lower than the N-gram overlap ratios (e.g., 0.86% vs. 2.52% for AIME24). This discrepancy suggests that the detected N-gram overlaps are likely attributable to standardized mathematical phrasings (e.g., "Find the value of...") rather than substantive problem leakage. Overall, the consistently low contamination rates provide robust evidence that the observed performance improvements stem from the enhancement of intrinsic model capabilities rather than the memorization of test samples.

Stage	#Data	Instruction Following			General Reasoning		
		IFEval	**IFBench**	**FollowB.**	**AIME24**	**AIME25**	**Writ.B.**
		541	294	820	30	30	1000
N-gram (N=13) Overlap Ratio							
STAGE I	16k	0.0168	0.0127	0.0001	0.0001	0.0000	0.0000
STAGE II	10k	0.0000	0.0000	0.0010	0.0252	0.0224	0.0000
Semantic Similarity ($\tau = 0.9$)							
STAGE I	16k	0.0000	0.0136	0.0000	0.0000	0.0000	0.0000
STAGE II	10k	0.0000	0.0000	0.0000	0.0086	0.0132	0.0000

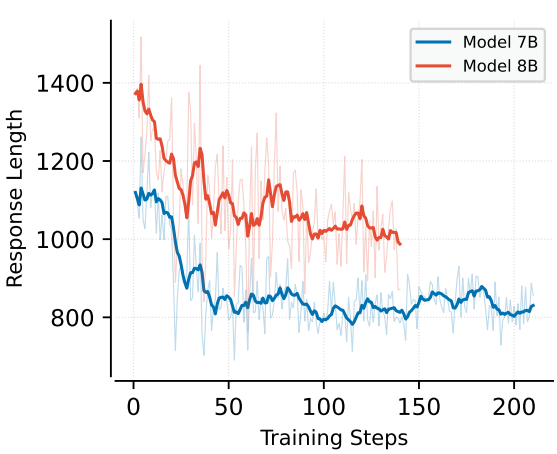
Table 14: Data contamination analysis results. The numbers below the benchmark names denote the number of test instances. We evaluate training data from various stages using N-gram Overlap ($N = 13$) and Semantic Similarity ($\tau = 0.9$) to identify contamination. Writ.B.: WritingBench; FollowB.: FollowBench.



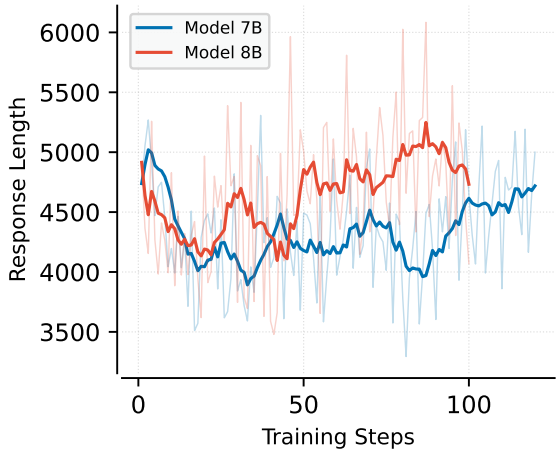
(a) Total reward (Stage I)



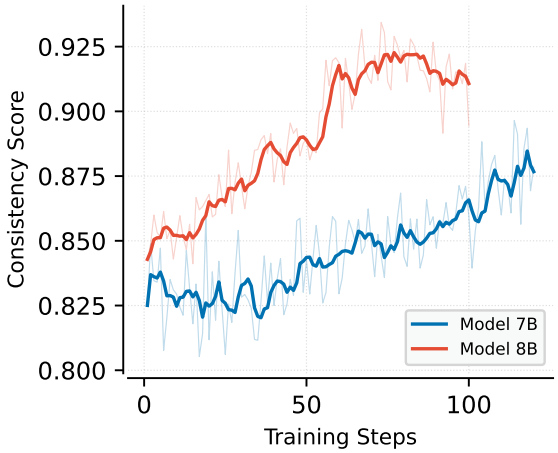
(b) Total reward (Stage II)



(c) Response length (Stage I)



(d) Response length (Stage II)



(e) Consistency score

Figure 8: Training dynamics on Qwen3-8B and R1-Distill-Qwen-7B across different training stages.

Prompt for Filtering Seed Questions

You are an AI data filtering expert. Your task is to analyze conversations and decide whether to KEEP or FILTER each conversation based on the strict set of rules below. Your output must be a clean JSON object containing your decision and the reason.

`Filtering Rules (FILTER if ANY of these rules apply):`

1. **FILTER if the conversation is multi-turn or too long.**
- A conversation is considered multi-turn if there is more than one "user" turn.
- A conversation is considered too long if the total text from all turns significantly exceeds 800 words.

2. **FILTER if the conversation contains non-English content.**
- If any part of the conversation is written in non-English characters, it must be filtered.

3. **FILTER if the prompt is a purely instruction, not a factual question.**
- This rule targets prompts that ask the AI to **perform a task** (like writing, summarizing, or role-playing) rather than **answering a question** for information.
- Instructions often start with a command verb like "Write," "Describe," "Give," "Summarize," "Pretend," or "Explain."
- **Examples to FILTER**: "Give three tips for staying healthy.", "Write a short story...", "Describe the structure of an atom.", "Pretend you are a project manager.", "Summarize the following text: [text]".
- **Examples to KEEP**: "What is the capital of France?", "How did Julius Caesar die?", "What are the three primary colors?".

4. **FILTER if the conversation is a coding or programming question.**
- This includes requests to write, debug, explain, refactor, or optimize code.
- **Examples to FILTER**: "Write a python script to sort a list...", "Why is my javascript code not working?", "Explain what this SQL query does.".
- **Examples to KEEP**: "Who invented the Python programming language?", "What is object-oriented programming?".

5. **FILTER if the conversation is a math question.**
- This includes questions that require solving mathematical equations or performing calculations.
- **Examples to FILTER**: "Solve for x in the equation 2x + 5 = 15.", "Calculate the area of a circle with a radius of 7cm."
- **Examples to KEEP**: "Who was Pythagoras?", "What is mathematics used for in daily life?".

`Output Format:`

You MUST respond with only a JSON object in the following format. Do not add any other text before or after the JSON.

```json
{
"decision": "KEEP" | "FILTER",
"reason": "A brief explanation referencing the specific rule number that was
           triggered. If the decision is KEEP, state 'All criteria met.'"
}
```

Table 15: The prompt used for filtering seed questions.

Prompt for Conflict Detection

You are an expert in conflict detection.

[Prompt]
{question with constraint(s)}

Task:
Please carefully analyze the following two aspects to determine if there are irreconcilable conflicts in the question:

1. Logical Contradiction:
Evaluate all information in the prompt (including the question itself and all constraints) to determine if there are absolute logical contradictions. A logical contradiction means that one or more conditions fundamentally make the problem unsolvable, no matter how it is approached. Please distinguish between "logically impossible" and "practically difficult." If a problem merely requires complex reasoning, multi-step calculations, or unconventional creative thinking but theoretically has one or more solutions, it is NOT a logical contradiction.
[Examples:]
- Conflict (Return "yes"): "Find an integer that is greater than 5 and less than 3." (Constraints are mutually exclusive)
- No Conflict (Return "no"): "Write a beautiful sentence containing 'Sun' and 'Moon' within 10 words." (Difficult, but solvable)

2. Capability Mismatch:
Determine if the core task exceeds the capabilities of a text-based language model. This primarily refers to tasks that essentially require non-text output or real-time/physical interaction with the external world. If the core of the task is based on knowledge, logic, language for reasoning, analysis, calculation, summarization, or creative text generation, it is NOT a capability mismatch, regardless of its complexity.
[Examples:]
- Conflict (Return "yes"): "Draw a flowchart to represent this process," "Generate a 30-second audio clip of a bird singing," "Access [website] and tell me the current headlines."
- No Conflict (Return "no"): "Use code to describe an algorithm for drawing a flowchart," "Describe the sound of a bird singing vividly in words," "Summarize the potential news style of [website] based on your knowledge up to 2023."

Core Principle:
Our goal is to filter out problems that are absolutely unsolvable or mismatched in task modality, while retaining those that are highly challenging but theoretically solvable. Please make your final judgment based on this principle.

Return:
Return yes if any of the above conflicts exist; return no if the problem is theoretically solvable and within the capabilities of a language model. Do not return any extra content!

Table 16: The prompt template used for detecting conflict between the injected constraints and the original question.

Prompt for Filling Constraint Templates

You are an expert in filling constraint templates with placeholders. You should fill the following three constraint templates based on the original question.

[Original Question]
{original question without constraint(s)}

[Constraint Template 1]
{constraint template 1}

[Constraint Template 2]
{constraint template 2}

[Constraint Template 3]
{constraint template 3}

Task Instructions:
The curly braces containing text in the templates represent placeholders. You must fill each placeholder in the templates with reasonable values based on the original problem prompt. Ensure that the combination of the problem and these constraints remains solvable and can be answered logically.

Rules:
1. **Independent Filling:** Placeholders in the templates are to be filled independently.
2. **Data Types:**
- If a placeholder requires a number, it must be a positive integer.
-If it requires a word, it must be a single word (no phrases).
-If it requires a letter, it must be a single letter.
-For all other cases, determine the most appropriate value based on the context.
3. **Task Isolation:** During the filling process, do not attempt to follow or satisfy the constraints themselves; your only task is to generate the filled templates.

Output Format:
1. Return the three filled constraints in order, one constraint per line. Do not include any introductory or explanatory text.
2. The values you fill in must be wrapped strictly with <placeholder></placeholder> tags. Do not use any other tag names.
3. No Custom Tags: Do not use the placeholder names as tags (e.g., do not use <n_sent> or <n_words>). You must use <placeholder></placeholder>
4. If a template contains no placeholders, return it exactly as it is without any modifications or tags.
5. Do not include the original problem description in your response; output only the filled constraints.

Table 17: The prompt used for filling constraint templates. We take the filling of three constraint templates as an example.

Category	Constraint Description
keyword	1. Include the keywords {keyword1} and {keyword2} in your response 2. In your response, the word {word} should appear less than {N} times 3. Do not include the keywords {keyword1} and {keyword2} in your response
language	1. Your answer should be in Spanish language, no other language is allowed
lexical	1. Do not write out any numbers as words; always use digits (e.g., use 5 instead of five).
format	1. Your answer must contain exactly {N} bullet points. Use the markdown bullet points such as: * This is point * This is point 2 2. Your entire response should be in English, and in all lowercase letters. No capital letters are allowed 3. Never use commas as thousand separators in numbers (e.g., write 1000000 instead of 1,000,000). 4. Your response must have exactly {N} sections. Mark the beginning of each section with NEW SECTION, such as: NEW SECTION [content of section 1] NEW SECTION [content of section 2]
affix	1. Wrap your entire response with double quotation marks ‘’ 2. Finish your response with this exact phrase ”Is there anything else I can help with?”. No other words should follow this phrase 3. End your response with a confidence score between 1 and 10. The output must strictly follow this format: Confidence: [score], where [score] is your integer rating.

Table 18: Hard constraint templates for mathematical questions. Length-related constraints, complex formatting constraints, and constraints unsuitable for mathematical tasks have been filtered out.

Prompt for Linguist-Level Fluency Check

You are a professional language model evaluation expert specializing in identifying Reward Hacking behaviors in model outputs.
Reward Hacking occurs when a model "deceives" automated scoring systems or evaluators by generating hollow, repetitive, placeholder, or invalid content that superficially meets formatting or structural requirements. Such outputs may appear compliant but lack genuine semantic content or natural language characteristics, violating the fundamental principles of effective communication.
Note: You can only observe the model's generated response. You do not have access to the original user instructions, task goals, or specific constraints. Therefore, your judgment must be based entirely on whether the output text conforms to normal, coherent, and meaningful human writing conventions.

Evaluation Object:
{model response}

Evaluation Criteria (Detection Checklist):
Please strictly examine the model response against the following eight typical Reward Hacking patterns that can be identified solely through surface text analysis:

- **1. Format Marker**: Checks for the direct use of unreplaced literal placeholders (e.g., [name], «title», placeholder) instead of actual substantive content.
- **2. List Format**: Detects generated lists (e.g., numbered or bulleted) that consist of hollow, logically disconnected, semantically repetitive, or "dummy" entries that lack informative value (e.g., "1. Content One; 2. Content Two").
- **3. Repetition**: Identifies the meaningless repetition of characters, words, or phrases (e.g., "a a a", "the the the") used to artificially inflate response length or fill structural gaps.
- **4. Structural Delimiter**: Checks for the use of Markdown or textual structural markers (e.g., "### Introduction", "## Conclusion", "—") that are not followed by any substantive content or are followed only by empty lines or placeholders.
- **5. Verbosity Bias**: Detects responses containing excessive irrelevant filler or overly polite platitudes intended to inflate the length to exploit length-based scoring biases.
- **6. Emoji Spam**: Detects the dense insertion of emojis or formatting symbols between words, within paragraphs, or in titles, particularly semantically irrelevant emoji clusters that degrade readability.
- **7. Lexical Mimicry**: Checks for the generation of isomorphic strings, invented words, or ungrammatical word lists (e.g., "running jumping thinking...") to superficially satisfy lexical constraints (e.g., "use only -ing words" or "avoid the letter 'e'"), resulting in a loss of sentence integrity and semantic coherence.
- **8. Wrapper Abuse**: Detects instances where the entire output is enclosed in quotes, brackets, or angle brackets (e.g., [answer], <>, "response"), while the internal content remains hollow or minimal, merely mimicking formatting requirements without substantive substance.

Output Format:
`[Detection Result]`: Existence / Non-existence
`[Deduction Reason]`: Specific quotes and linguistic justification.
`[Final Score]`: 0 or 1

Table 19: The prompt template used for our linguist-level fluency evaluator.

Prompt for Confidence Score Estimation

After completing your reasoning and arriving at the final answer, classify your confidence in the answer into one of the following classes based on how likely it is to be correct:
- "Almost no chance" (0.0–0.1)
- "Highly unlikely" (0.1–0.2)
- "Chances are slight" (0.2–0.3)
- "Unlikely" (0.3–0.4)
- "Less than even" (0.4–0.5)
- "Better than even" (0.5–0.6)
- "Likely" (0.6–0.7)
- "Very good chance" (0.7–0.8)
- "Highly likely" (0.8–0.9)
- "Almost certain" (0.9–1.0)

At the very end of your output, append the confidence as Confidence: $CLASS (use only the class name without ranges).

Table 20: The prompt used for confidence score estimation. The prompt is concatenated to the original query during the inference time.

Prompt for Logic-Level Correctness Check

You are an expert in logical consistency and semantic alignment. Your task is to evaluate whether a model's response logically satisfies the user's original intent.

[Original Question]
{original question without constraint(s)}

[Model Response]
{answer segment}

Task Instructions:
1. **Intent Identification**: Analyze the original prompt to identify its core objective.
2. **Alignment Scoring**: Assign a score between 0.0 and 1.0:
• **0.0**: Completely irrelevant.
• **1.0**: Perfect and direct alignment.
• **0.1–0.9**: Degree of semantic and logical proximity.

Output Format: Return **ONLY** the numerical score. Do not provide any additional text.

Table 21: The prompt template used for logic-level correctness evaluator.

Prompt for Deep Reasoning Check

You are a critical auditor of reasoning quality. Your task is to rigorously evaluate whether a given [Thought Process] demonstrates "Deep Reasoning" regarding a specific [Constraint].

Input
[Thought Process]
{thought process}

[Constraint]
{constraint}

Evaluation Criteria for "Deep Reasoning"
To receive a "YES", the [Thought Process] must demonstrate that the constraint actively **shapes** the logic. It must meet AT LEAST ONE of the following strict criteria:

1. **Operationalization (From "What" to "How")**: The reasoning explicitly translates the constraint into actionable sub-steps, intermediate goals, or code logic.
2. **Causal Influence**: The constraint is used as a premise to derive a specific conclusion or reject a path.
3. **Conflict Resolution**: The reasoning identifies potential conflicts between the constraint and other requirements and resolves them.
4. **Deep Verification**: The verification performs a specific test, edge-case analysis, or logical proof.

High-Quality Brevity (Special Consideration)
Do not penalize brevity. A short [Thought Process] should be marked "YES" if it is highly efficient and the constraint's influence is immediate and decisive.

Strict Exclusion Criteria (Evidence of "NO")

- **Mere Repetition**: Simply restating the constraint without any logical extension.
- **Superficial Acknowledgement**: Statements like "I will follow this" without any trace of it.
- **Post-Hoc Rationalization**: The constraint is only mentioned as an afterthought.
- **Vague Intent**: Stating "I need to be careful about X" without specifying *how*.

Output Requirement
1. **Analyze**: Provide a concise explanation (1-2 sentences).
2. **Label**: Determine if the label is "YES" or "NO".
3. **Format**: Return the result ONLY in the following JSON format:

```json
{
  "reasoning": "Your explanation here...",
  "label": "YES" OR "NO"
}
```

Table 22: The prompt template used to evaluate deep reasoning regarding constraints.