Logic-RL: Unleashing LLM Reasoning with Rule-Based Reinforcement Learning

Liyu Yang
2024134030
yangly2024@shanghaitech.edu.cn

Lingkai Zu
2024233244
zulk2024@shanghaitech.edu.cn

Xiyue Peng
2024233182
pengxy2024@shanghaitech.edu.cn

Abstract

In this project, we explored the reasoning capabilities of small-scale language models based on the *Logic-RL: Unleashing LLM Reasoning with Rule-Based Reinforcement Learning* framework. We attempted to align these models with reasoning-related downstream tasks through various post-training algorithms. Our experiments revealed that the small models failed to effectively learn reasoning—instead, they often abandoned logical thinking altogether. We hypothesize that this limitation stems from either insufficient model capacity to support reasoning or the need for more specialized reward signals to elicit their latent reasoning abilities.

1 Introduction

Although large language models have demonstrated remarkable performance in mathematical reasoning and code generation, the more successful cases, such as DeepSeek-R1[3], Kimi-K1.5[12], and OpenAI-o1, are all large-scale models supported by substantial computational resources. This has confined the development and application of LLMs primarily to a handful of tech giants and research institutions with massive budgets. This raises the question of how we can effectively pursue large model research within the constraints of limited computing power.

Aligning smaller-scale models offers a promising approach to investigating language model alignment when computational resources are limited. The alignment study of smaller-scale models is of considerable research significance. Smaller-scale models have specific application scenarios, particularly on edge devices such as IoT devices, smartphones, and embedded systems that operate under severe resource constraints (limited computing power, storage, and battery life). While these devices cannot support large-scale LLMs, smaller-scale models can be effectively deployed to enable localized, low-latency, and privacy-preserving intelligent processing (e.g., voice assistants, text summarization, and translation). Additionally, smaller-scale models exhibit much lower energy consumption compared to LLMs, which resonates with current priorities in sustainable computing and carbon footprint reduction.

Therefore, we will further explore the issues mentioned during the reproduction of DeepSeek-R1 in *Logic-RL: Unleashing LLM Reasoning with Rule-Based Reinforcement Learning* [14]:

Can similar reasoning abilities emerge in smaller-scale models?

The achievement of the 7B model raises the question of whether alignment techniques can replicate such impressive dynamic reasoning performance on smaller-scale models. However, smaller-scale

Preprint. Under review.

models exhibit diminished expressive and inferential abilities relative to larger models, stemming from their reduced parameter size. Therefore, we conduct experiments on a 0.5B model to compare existing LLM post-training methods and discuss the seven experimental questions proposed in Xie et al. [14]. Overall, our contribution to this project is as follows,

- We utilized the Synthetic Logic Puzzles Framework to train a 0.5B-scale model on reasoning tasks.
- Compared to Logic-RL [14], we consider a broader range of post-training algorithms.
- Our experiments revealed the challenge of limited reasoning performance in 0.5B-scale models, and we considered possible ways to improve it.

2 Synthetic Logic Puzzles Framework

2.1 Data Synthesis

To answer our core question—*"Can a **0.5 B** model, trained under a single-GPU budget, acquire the dynamic reasoning that DeepSeek-R1 (7 B) exhibits?"*—we adopt the **Synthetic Knights & Knaves (K&K) logic-puzzle suite** introduced by Xie et al. [13]. The *Knights & Knaves* (K&K) dataset [13] is an *algorithmically generated* suite of logical-reasoning puzzles in which every inhabitant is either a knight (truth-teller) or a knave (liar). The solver must deduce each character's identity from their utterances. K&K is especially attractive for small-model alignment because it is **highly controllable**:

(1) **Procedural generation.** Puzzles are created with formal logic templates, ensuring internal consistency while providing essentially infinite, *unseen* test cases for generalisation.

(2) **Adjustable difficulty.** Complexity is tuned by the *number of speakers* (2–8) and the *depth of Boolean operators* (one to four nested connectives). This enables curriculum learning and out-of-distribution (OOD) evaluation—easier instances train the model, harder ones probe generalisation.

(3) **Deterministic verification.** Each puzzle admits a *single* ground-truth assignment, guaranteed by the generator. Solutions can therefore be evaluated automatically and unambiguously, virtually eliminating reward-hacking through ambiguous text.

Example.

> **Zoey**: "Oliver is not a knight."
> **Oliver**: "Oliver is a knight *iff* Zoey is a knave."
>
> **Goal**: Identify who is a knight and who is a knave.

Ground truth: (1) Zoey is a **knave**; (2) Oliver is a **knight**. Because the generator stores this label, evaluation reduces to pattern matching rather than expensive external calls.

2.2 Rule-Based Reward Modelling

In reinforcement learning, the reward is the sole optimisation signal. Following Xie et al. [14], we design a *two-component, rule-based* reward that is both lightweight and robust to hacking:

$$R \;=\; S_{\text{format}} + S_{\text{answer}}.$$

(i) Format reward. The agent must answer in the form `<think> reasoning </think><answer> conclusion </answer>`. Using regular expressions, we check that:

- each tag appears *exactly once* and in the correct order,
- the `<think>` block contains non-trivial reasoning, and
- the `<answer>` block contains an extractable list of identities.

The score is assigned as

$$S_{\text{format}} = \begin{cases} +1 & \text{format correct,} \\ -1 & \text{otherwise.} \end{cases}$$

(ii) Answer reward. After validating the format, we compare the extracted conclusion with the ground truth:

$$S_{\text{answer}} = \begin{cases} +2 & \text{fully correct,} \\ -1.5 & \text{partially correct,} \\ -2 & \text{unparsable or missing.} \end{cases}$$

Why only two rewards? Early experiments revealed multiple hacking behaviours—e.g. omitting `<think>`, hiding reasoning inside `<answer>`, or padding irrelevant text. By strictly separating *structure* (format) from *content* (answer) and making each component coarse but decisive, we obtain a nearly unhackable signal that still fits easily within a single-GPU budget.

System prompt. The system prompt during training is

```
You are a helpful assistant. The assistant first thinks about the
reasoning process in the mind and then provides the user with the answer.
The reasoning process and answer are enclosed within <think> </think>
and<answer> </answer> tags, respectively, i.e., <think> reasoning process
here </think><answer> answer here </answer>. Now the user asks you to
solve a logical reasoning problem. After thinking, when you finally reach
a conclusion, clearly state the identity of each character within <answer>
</answer> tags. i.e., <answer> (1) Zoey is a knight, (2) ... </answer>.
```

And then we will give the current question. Note that there is a fixed instruction ending with a literal `<think>` token, which greatly improves the base model's compliance with the required tag syntax before RL begins.

3 Post Training Method Overview

As illustrated in Figure 1, the fine-tuning of large language models primarily involves three key steps: (1) supervised fine-tuning (SFT), (2) training a reward model, and (3) fine-tuning via reinforcement learning (RL) algorithms [6]. Among these, the third step, fine-tuning using RL algorithms, serves as

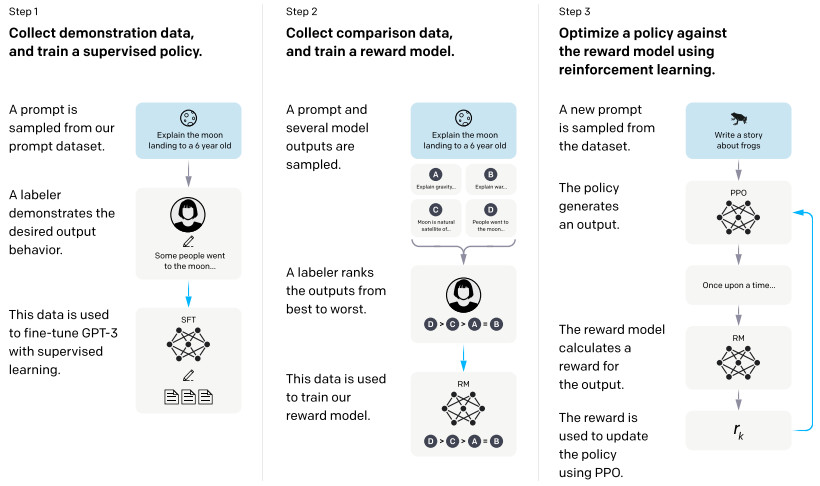

Figure 1: The pipeline of reinforcement learning from human feedback (RLHF) [6].

the core of the entire RLHF pipeline. Compared to the next-token prediction in SFT, this approach enables holistic semantic control over the generated text, resulting in outputs that exhibit better coherence and emotional consistency, thereby achieving performance that is more aligned with downstream tasks. We refer to this type of training, which aims to better align the model with

downstream tasks, as "post-training". Furthermore, compared to direct preference optimization (DPO) [7] based on Bradley-Terry reward models [2], the RL framework demonstrates greater adaptability to diverse reward functions, such as the rule-based reward considered in this study.

Remark 1. *As described in Ziegler et al. [15], Ouyang et al. [6], the generation process of an LLM can be framed as a Markov decision process (MDP). Starting from the initial state s_0, the language model π_θ outputs a token a_h at each step from the vocabulary set, forming a new state $s_h = (s_0, a_1, a_2, \ldots, a_{h-1}, a_h)$. The generation process concludes when a specific end token is produced or the maximum length H is reached, with the final response denoted as y. The reward function $R(x, y)$ learned in the previous stage is used to evaluate the quality of the response y. Therefore, the objective of reinforcement learning fine-tuning is to maximize the (regularized) reward as follows:*

$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(y|x)} \left[R(x, y)\right] - \beta\mathbb{KL}\left(\pi_\theta \| \pi_{ref}\right)$$

The reward model $R(x, y)$ is trained before and frozen in this step. The regularized term $\beta\mathbb{KL}\left(\pi_\theta \| \pi_{ref}\right)$ ($\beta \geq 0$) is to prevent the fine-tuned model from diverging too far from the reference model and to avoid over-optimization of the (possibly inaccurate) reward model.

The Proximal Policy Optimization (PPO) algorithm, as the state-of-the-art (SOTA) method in modern RL, was the first choice for the post-training step illustrated in Figure 1. However, the PPO algorithm requires the assistance of a value model for policy evaluation, and this value model typically has a scale comparable to the policy network itself. In the context of LLM fine-tuning, this usually constitutes a significant computational overhead. Therefore, Li et al. [5] and Shao et al. [9] proposed rethinking the classic REINFORCE framework [11, 10] to avoid the heavy GPU memory overhead introduced by the value model. Reviewing the REINFORCE algorithm, its core mechanism involves computing the policy gradient through Monte Carlo sampling, which has the form

$$g = \mathbb{E}\left[\sum_{h=0}^{H} \psi_h \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right],$$

where ψ_h is the value used to evaluate the current policy. A naive choice would be to use the return as ψ_h directly, but it often leads to substantial approximation errors that can degrade policy performance significantly. In PPO, the advantage function is used as ψ_h, which explains why PPO requires the introduction of a value model. To avoid introducing a value model while also reducing variance, a viable approach is to use the return minus a baseline b, i.e., a "surrogate advantage." This leads us to the key distinction among the post-training algorithms we will discuss subsequently: how to design the baseline function? The pipeline of estimating the "surrogate advantages" of different algorithms is shown in Figure 2, and we describe their details in the following.

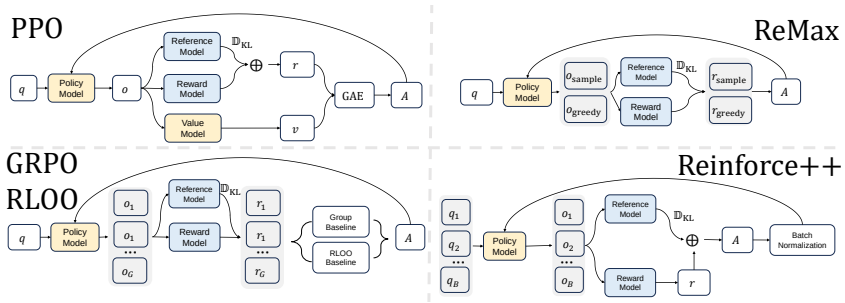

Figure 2: The pipeline of these algorithms [4].

ReMax [5]. During the RLHF generation process, we can scale the stochasticity of policy sampling using the temperature parameter. Li et al. [5] propose using a deterministic greedy policy as a baseline for gradient estimation. Specifically, the policy gradient is estimated using

$$\hat{g} = \mathbb{E}\left[\sum_{h=0}^{H} (r(x^i, a^i_{1:h-1}) - r(x^i, \bar{a}^i_{1:h-1})) \nabla_\theta \log \pi_\theta(a_h \mid x^i, a_{i:h-1})\right].$$

Interestingly, this method of reducing gradient estimation variance by leveraging greedy strategies has also been employed in image captioning [8]. Since the responses obtained through the greedy

strategy are independent of the current response, the gradient estimation in the ReMax method is unbiased.

GRPO [9]. Unlike ReMax, which employs a greedy strategy, GRPO retains the PPO-style approach but modifies the definition of the advantage function. Unlike PPO which uses value network predictions, GRPO generates multiple responses for the same prompt and analyzes the advantage of each generation - that is, it defines advantage as

$$A_{i,h} = \frac{r_i - \mathrm{mean(r)}}{\mathrm{std(r)}},$$

where r is the vector composed by the reward of different responses from the same prompt.

RLOO [1]. The RLOO algorithm is similar to the GRPO, but the intuition behind is the leave one out estimator as

$$\frac{1}{k}\sum_{i=1}^{k}[R(y_{(i)}, x) - \frac{1}{k-1}\sum_{j\neq i} R(y_{(j)}, x)]\nabla \log \pi(y_{(i)} \mid x) \text{ for } y_{(1)}, \cdots, y_{(k)} \sim \pi_\theta(\cdot \mid x).$$

Since RLOO considers a baseline unrelated to the current sample, RLOO is also an unbiased estimator.

REINFORCE++ [4]. REINFORCE++ is the original reinforce frame work

$$\sum_{h=1}^{H} r(x^i, a_{1:h})\nabla \log \pi(x, a_{1:h})$$

with a batch normalize.

4 Experiments Results And Discussion

In this section, we discuss and analyze the question "*Can similar reasoning abilities emerge in smaller-scale models?*" by examining the experimental results in conjunction with the seven research questions proposed in Xie et al. [14].

4.1 Experimental Settings

Training Settings. We use Qwen2.5-0.5B-Instruct[1] as the initial model for post-training and train it on the entire Knights and Knaves (K&K) puzzles[13] dataset. We use the same training parameters as shown in Table 1 for the algorithm discussed in Section 3. And all the training processes are finished on one NVIDIA L40S.

Table 1: Important Training Parameters.

Algorithm	GRPO	ReMax	REINFORCE++	RLOO
Training Batch Size			2	
Rollout N			8	
KL Coefficient			0.001	
Max Response Length			2048	
Learning Rate			3×10^{-7}	

Inference Settings. Similarly, we conduct inference on a single NVIDIA L40S GPU, with a maximum response length of 2048 tokens and an inference temperature set to 1.0.

4.2 Consumption Resource Analysis

We first evaluate the computational cost incurred by different algorithms. In RQ1 of Xie et al. [14], the computational overhead relationships of GRPO, REINFORCE++, and PPO in terms of single-step time were analyzed, demonstrating that PPO is 138% slower than REINFORCE++ and GRPO. In this project, we conducted a detailed analysis of the computational costs incurred by the four training methods.

[1] `https://huggingface.co/Qwen/Qwen2.5-0.5B-Instruct`

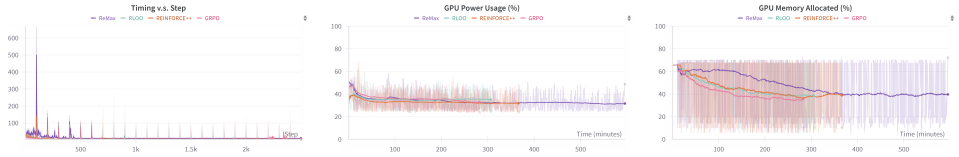

Figure 3: Consumption resource analysis regarding training speed (left), GPU power usage (middle), and GPU memory allocated (right).

From the results of Figure 3, we observed that consistent with our analysis in Section 3, the single-step training time and GPU computational resource overhead of GRPO, ReMax, REINFORCE++, and RLOO are within our acceptable range. Compared to the other three algorithms, ReMax requires generating an additional greedy response, resulting in slightly longer training time. However, these four algorithms are remarkably acceptable in terms of computational resource demands and time overheads, especially when compared to PPO's requirements under limited computing resources.

4.3 Reasoning Performance Analysis

Although the 0.5B model showed promising computational efficiency, its reasoning capabilities fell short of expectations. Following the reward structure of the synthetic logic puzzles framework discussed in Section 2, we analyze the performance of aligned models from two perspectives: format reward and answer reward.

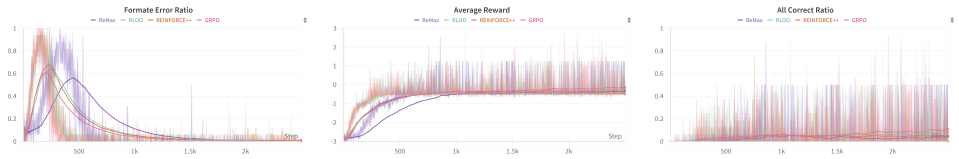

Figure 4: Reasoning performance analysis based on formate error ratio (left), average reward (middle), and all correct ratio (right).

As evidenced by the training results in Figure 4, all algorithms successfully acquire the intended `<think>...<\think><answer> ...<\answer>` format and put the final answer in `<answer> ...<\answer>`. However, judging from the total rewards obtained and the overall accuracy, these four algorithms did not achieve satisfactory results. The evaluation results on the test set, as shown in Figure 5, further demonstrate that although the fine-tuned models have acquired certain reasoning capabilities compared to the initial model, these abilities are still far from satisfactory. Even GRPO only achieves an 11.72% all correct ratio on the test set, which is the best-performing algorithm. Next, we try to analyze the reasons behind the models' poor reasoning capabilities based on the performance during the experimental process.

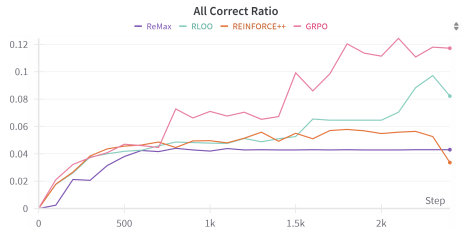

Figure 5: The overall performance of the four algorithms on the test set. The overall performance requires both format reward and answer reward.

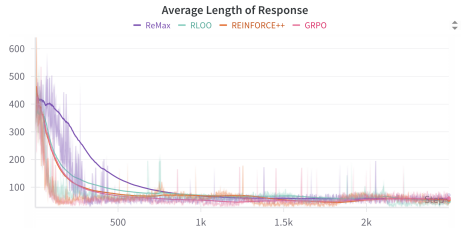

Figure 6: The average length of the responses during the training.

Is Format Reward a Blessing or a Curse? Although in Figure 4, we observe that formate can almost guarantee correctness as training progresses, as shown in Figure 6, the length of the text does not

increase further. Although Xie et al. [14]'s RQ7 mentions that there is no clear correlation between the length of generated text and the correctness of answers, this argument becomes meaningless if language models do not engage in any reasoning process at all. Additionally, as seen from the sampled examples in Table 2, the model's reasoning process appears to be largely superficial or fabricated. This leads us to question whether the approach proposed in [], using formatting patterns to

Table 2: The sample Responses.

Quiz	A very special island is inhabited only by knights and knaves. Knights always tell the truth, and knaves always lie. You meet 3 inhabitants: Scarlett, David, and Liam. According to Scarlett, "Liam is a knave if and only if Liam is a knight." David stated, "If Scarlett is a knave then Liam is a knave." Liam stated, "Scarlett is not a knave." So who is a knight and who is a knave?
Algorithm	**Response**
GRPO	<think>Scarlett is a knight, David is a knight, Liam is a knight </think><answer> Scarlett is a knight, David is a knight, Liam is a knight </answer>
ReMax	<think>Reasoning process: </think><answer> (1) Scarlett : knave, (2) David : knight, (3) Liam : knave </answer>
REINFORCE++	<think>Identify doubts </think><answer> penelope is a knave, aiden is a knight, amelia is a knave </answer>
RLOO	<think></think><answer> (1) Penelope is a knight, Aiden is a knight, Amelia is a knight. </answer>

guide model reasoning, is effective for small-scale models. Moreover, due to the absence of reasoning processes, we are also unable to observe the "Aha Moment" during the training process. This presents an interesting direction for future research. Regarding this issue, we have two levels of consideration:

- Whether small-scale models possess reasoning capabilities?
- What kind of rewards should we design to elicit reasoning from small-scale models?

Inspired by RQ2 in Xie et al. [14], we investigate whether we can stimulate a model's latent reasoning ability by identifying and rewarding certain thinking tokens, or by providing high-quality supervision over thought chains to guide efficient reasoning in small-scale models.

5 Conclusion

Overall, in this project, we aimed to further investigate the question of "*Can similar reasoning abilities emerge in smaller-scale models?*" by applying various post-training algorithms to train Qwen-2.5-0.5B within the Synthetic Logic Puzzles Framework. While we achieved promising results in terms of training efficiency, the model's reasoning capabilities fell short of our expectations. Through analysis, we identified potential issues and proposed possible solutions, offering viable optimization pathways for training smaller-scale models.

6 Team Member Responsibilities

This section outlines the specific responsibilities assigned to each team member for this project.

Liyu Yang. Primarily responsible for analyzing literature and writing the Section 2 and Section 3 of the paper.

Linkai Zu. Assisted in literature analysis and code experiments, and was responsible for writing Section 1 as well as designing the poster.

Xiyue Peng. Responsible for conducting code experiments, writing Section 4 and the final presentation and demonstration.

Overall, we consider the contributions of all three team members to be equal in this project.

References

[1] Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *CoRR*, 2024.

[2] Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

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

[4] Jian Hu. Reinforce++: A simple and efficient approach for aligning large language models. *arXiv e-prints*, pages arXiv–2501, 2025.

[5] Ziniu Li, Tian Xu, Yushun Zhang, Zhihang Lin, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. Remax: A simple, effective, and efficient reinforcement learning method for aligning large language models. *arXiv e-prints*, pages arXiv–2310, 2023.

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

[7] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.

[8] Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7008–7024, 2017.

[9] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[10] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[11] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[12] Kimi Team, A Du, B Gao, B Xing, C Jiang, C Chen, C Li, C Xiao, C Du, C Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms, 2025. *URL https://arxiv. org/abs/2501.12599*, 2025.

[13] Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. On memorization of large language models in logical reasoning. 2024. *URL https://arxiv. org/abs/2410.23123*.

[14] Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*, 2025.

[15] Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.