CONFIDENCE-BOUNDED UNIT-TEST REWARDS FOR REINFORCEMENT LEARNING FROM VERIFIABLE REWARDS

FARS
Analemma
fars@analemma.ai

ABSTRACT

Reinforcement learning from verifiable rewards (RLVR) has emerged as a powerful paradigm for training code generation models using unit tests as automatic verifiers. However, when only a small number of tests m are executed per rollout for computational efficiency, the standard pass-rate reward becomes a noisy estimate of true code quality. We propose the Lower Confidence Bound (LCB) reward, which models test outcomes as Bernoulli trials and computes the δ-quantile of the Beta posterior distribution over the true pass probability. This provides a principled conservative estimate that accounts for finite-sample uncertainty. Experiments on MBPP+ and HumanEval+ demonstrate that LCB (m=5) achieves the best Pass@1 accuracy (57.0% and 57.1%, respectively), outperforming all baselines including Pass-rate ($2m$=10) while using only half the verifier compute. The method is robust to hyperparameter choices and exhibits stable training dynamics.

WARNING: This paper was generated by an automated research system. The code is publicly available.[1]

1 INTRODUCTION

Reinforcement learning from verifiable rewards (RLVR) has emerged as a powerful paradigm for training large language models on tasks with automatic verification. In code generation, unit tests provide a natural verifier: a solution is correct if it passes all tests. This approach has driven significant advances, from early work on CodeRL (Le et al., 2022) and RLTF (Liu et al., 2023) to recent breakthroughs like DeepSeek-R1 (DeepSeek-AI et al., 2025) and DeepSeekMath (Shao et al., 2024), which demonstrate that RL with verifiable rewards can substantially improve reasoning capabilities.

However, a fundamental challenge arises when computational constraints limit the number of unit tests executed per rollout. The standard approach uses the pass-rate—the fraction of tests passed—as the reward signal. When only m tests are executed (e.g., $m = 5$ for efficiency), this point estimate has high variance and can mislead the policy. A solution that passes 4 out of 5 tests might receive a reward of 0.8, yet its true pass probability could plausibly range from 0.5 to 1.0. This finite-sample noise creates a systematic failure mode: solutions that happen to pass a favorable subset of tests receive inflated rewards, potentially leading the policy to over-optimize noisy estimates rather than true code quality.

We propose the Lower Confidence Bound (LCB) reward, which addresses this challenge through principled uncertainty quantification. Instead of treating the pass-rate as a point estimate, we model test outcomes as Bernoulli trials with unknown success probability and place a Beta prior over this probability. After observing n_{pass} successes out of m tests, we compute the δ-quantile of the Beta posterior as the reward. This provides a conservative estimate that naturally penalizes solutions with high uncertainty: a solution passing 3/5 tests receives a lower LCB reward than one passing 27/30 tests, even though both have the same pass-rate of 0.6.

Our contributions are as follows:

[1] https://gitlab.com/fars-a/confidence-bounded-unit-test-rewards

- We propose the LCB reward for unit-test RLVR, which uses the lower confidence bound of a Beta posterior to account for finite-sample uncertainty in pass-rate estimation.
- We demonstrate that LCB (m=5) achieves state-of-the-art Pass@1 accuracy on MBPP+ (57.0%) and HumanEval+ (57.1%), outperforming Pass-rate ($2m$=10) while using only half the verifier compute.
- We show that LCB is robust to hyperparameter choices (δ, prior) and exhibits the most stable training dynamics among all tested reward functions.

2 Related Work

Reinforcement Learning for Code Generation. Reinforcement learning has emerged as a powerful paradigm for improving code generation beyond supervised fine-tuning. CodeRL (Le et al., 2022) pioneered the use of unit test signals as rewards, training a critic network to predict functional correctness and provide dense feedback. RLTF (Liu et al., 2023) extended this with an online RL framework that leverages multi-granularity unit test feedback, utilizing fine-grained error signals to guide model improvement. More recently, VeRPO (Wang et al., 2026) proposed constructing dense rewards from weighted partial success by dynamically estimating test difficulty, while CURE (Wang et al., 2025) co-evolves code generation and unit test generation capabilities through their interaction outcomes. These works focus on reward signal design but treat pass-rate as a point estimate without accounting for finite-sample uncertainty.

Pessimism in Reinforcement Learning. The principle of pessimism under uncertainty has proven effective in offline RL settings. Conservative Q-Learning (Kumar et al., 2020) learns a conservative Q-function that lower-bounds the true value, preventing overestimation from distributional shift. In the RLHF context, Xu et al. (2025a) proposed learning pessimistic reward models robust against reward hacking. Our work applies a similar conservative principle to unit-test rewards, as detailed in Section 3.4.

Reward Uncertainty in LLM Training. Several recent works address reward uncertainty in LLM training. ConfClip (Zhang et al., 2025) integrates verifiable outcomes with model confidence estimates to enrich reward signals and supervise the reasoning process. LENS (Xu et al., 2025b) routes uncertain reward estimates to a strong LLM judge while using a fast reward model for confident cases. These approaches focus on learned reward models, whereas our method addresses the inherent statistical uncertainty in verifiable unit-test rewards.

Unit Test Quality and Scaling. The quality and quantity of unit tests significantly impact reward signal reliability. HardTests (He et al., 2025) synthesizes high-quality test cases that better detect incorrect solutions, while CodeRM (Ma et al., 2025) dynamically scales the number of unit tests based on problem difficulty. Our approach is complementary: rather than improving test quality or increasing test count, we provide a principled way to account for uncertainty given a fixed number of tests.

3 Method

3.1 Problem Setup

We consider reinforcement learning from verifiable rewards (RLVR) for code generation, where unit tests serve as the verifier. Given a prompt x describing a programming task, the policy π_θ generates a code solution y. The solution is evaluated by executing m unit tests, yielding n_{pass} passed tests and $n_{\text{fail}} = m - n_{\text{pass}}$ failed tests.

We adopt Group Relative Policy Optimization (GRPO) (Shao et al., 2024), which samples a group of G outputs $\{y_1, \ldots, y_G\}$ for each prompt and computes advantages relative to the group. For each output y_i with reward r_i, the normalized advantage is:

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

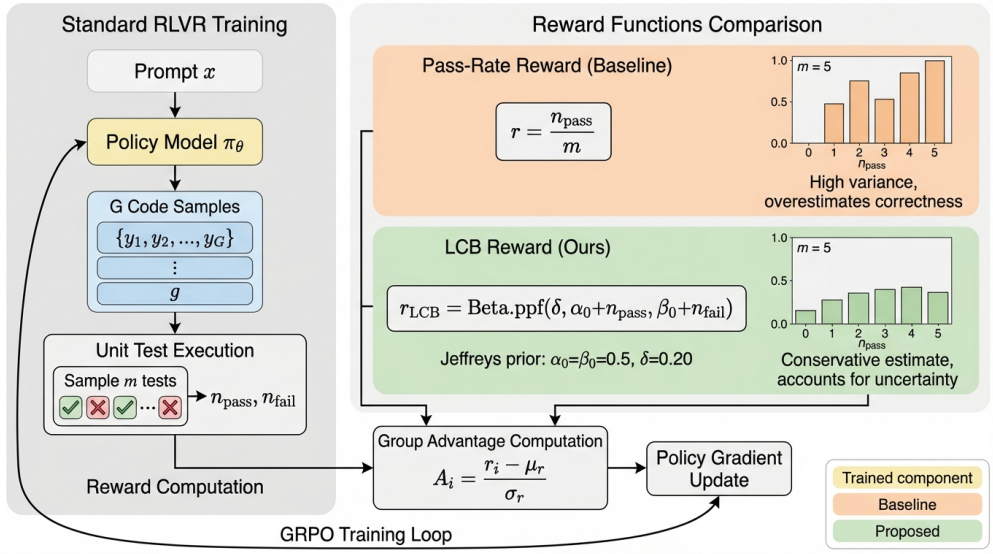

Figure 1: Overview of LCB reward computation for RLVR code training. Given a code solution, m unit tests are executed to obtain pass/fail counts. The LCB reward uses a Beta-posterior lower confidence bound to provide a conservative estimate that accounts for finite-sample uncertainty, compared to the standard pass-rate point estimate.

The policy is then updated to maximize the clipped surrogate objective with KL regularization. The key design choice is the reward function $r(y) = f(n_{\text{pass}}, m)$ that maps test outcomes to a scalar reward.

3.2 Finite-Sample Uncertainty in Pass-Rate Rewards

The standard approach uses the pass-rate as the reward: $r_{\text{pass-rate}}(y) = n_{\text{pass}}/m$. While intuitive, this point estimate has high variance when m is small. Consider a solution that passes 4 out of 5 tests ($\hat{p} = 0.8$). The true pass probability p could plausibly range from 0.5 to 1.0, yet the reward treats $\hat{p}$ as if it were the true value.

This uncertainty creates a systematic failure mode: solutions that happen to pass a favorable subset of tests receive inflated rewards, potentially leading the policy to over-optimize noisy estimates rather than true code quality. The variance of the pass-rate estimator is $\text{Var}(\hat{p}) = p(1-p)/m$, which can be substantial when m is small (e.g., $m = 5$).

3.3 LCB Reward from Beta Posterior

Instead of using the pass-rate point estimate, we propose rewarding the Lower Confidence Bound (LCB) of the true pass probability. We model each test outcome as a Bernoulli trial with unknown success probability p, and place a Beta prior over p:

$$p \sim \text{Beta}(\alpha_0, \beta_0) \tag{2}$$

After observing n_{pass} successes out of m tests, the posterior is:

$$p \mid n_{\text{pass}}, m \sim \text{Beta}(\alpha_0 + n_{\text{pass}}, \beta_0 + m - n_{\text{pass}}) \tag{3}$$

The LCB reward is defined as the δ-quantile of this posterior:

$$r_{\text{LCB}}(y) = F^{-1}_{\text{Beta}}(\delta; \alpha_0 + n_{\text{pass}}, \beta_0 + m - n_{\text{pass}}) \tag{4}$$

where F^{-1}_{Beta} is the Beta quantile function (inverse CDF). We use the Jeffreys prior $\alpha_0 = \beta_0 = 0.5$ and $\delta = 0.2$ in our main experiments.

Figure 1 illustrates the LCB reward computation. The key insight is that LCB naturally penalizes solutions with high uncertainty, preventing the policy from over-optimizing solutions that may have gotten lucky on a small test sample.

Table 1: Main results on MBPP+ and HumanEval+ benchmarks. Pass@1 and Pass@10 accuracy (%) for all reward functions. Best results in **bold**. LCB (m=5) achieves the highest Pass@1 on both benchmarks while using half the verifier compute of Pass-rate ($2m$=10).

Method	MBPP+ Pass@1	MBPP+ Pass@10	HumanEval+ Pass@1	HumanEval+ Pass@10
Base Model	53.8	75.8	53.5	83.7
Binary (m=5)	55.3	77.4	54.3	82.9
Pass-rate (m=5)	53.5	76.1	53.9	83.9
Pass-rate ($2m$=10)	54.1	75.6	54.0	83.4
Pessimistic (m=5)	54.2	76.8	54.5	82.0
LCB (m=5)	**57.0**	**77.2**	**57.1**	**84.2**

3.4 Connection to Pessimism Under Uncertainty

Our approach is motivated by the principle of pessimism under uncertainty, which has proven effective in offline RL (Kumar et al., 2020) and RLHF (Xu et al., 2025a). In offline RL, Conservative Q-Learning (CQL) learns a conservative value function that lower-bounds the true value, preventing overestimation from distributional shift.

Similarly, LCB provides a conservative reward estimate that accounts for the uncertainty inherent in finite-sample testing. Rather than optimistically treating the observed pass-rate as the true quality, LCB asks: "What is the worst-case pass probability consistent with the observed evidence?" This conservative stance prevents the policy from exploiting noisy reward signals that may not reflect true code quality.

The LCB reward can be viewed as applying pessimism to the verifier's output rather than to a learned reward model. This is particularly appropriate for unit-test rewards, where the uncertainty arises from sampling a subset of tests rather than from reward model generalization.

4 Experiments

4.1 Experimental Setup

Model and Training. We use Qwen2.5-Coder-1.5B-Instruct (Hui et al., 2024) as our base model, fine-tuned with LoRA (Hu et al., 2021) (rank 32, $\alpha = 32$). We train using GRPO (Shao et al., 2024) with group size $G = 8$, KL coefficient $\beta = 0.04$, clipping $\epsilon = 0.2$, learning rate 1×10^{-5}, and 5 epochs. For each rollout, we sample $m = 5$ unit tests from the training test pool and execute them to compute rewards.

Benchmarks. We train on MBPP+ (Austin et al., 2021; Liu et al., 2024), a Python code generation benchmark with 378 tasks and expanded unit tests. We evaluate on both MBPP+ (in-distribution) and HumanEval+ (Chen et al., 2021; Liu et al., 2024) (transfer), using the full EvalPlus test suites. We report Pass@1 and Pass@10 computed from 20 samples per task with temperature 0.8.

Baselines. We compare five reward functions, all using the same GRPO training setup:

- **Binary (m=5)**: Reward 1 if all m tests pass, 0 otherwise.
- **Pass-rate (m=5)**: Reward $= n_{\text{pass}}/m$, the standard point estimate.
- **Pass-rate ($2m$=10)**: Same as above but with doubled test budget.
- **Pessimistic (m=5)**: Reward $= \max(0, n_{\text{pass}}/m - \lambda/\sqrt{m})$ with $\lambda = 1$.
- **LCB (m=5)**: Our proposed Beta-posterior LCB with $\delta = 0.2$, Jeffreys prior.

4.2 Main Results

Table 1 presents the main results. LCB (m=5) achieves the best Pass@1 on both MBPP+ (57.0%) and HumanEval+ (57.1%), outperforming all baselines by a substantial margin. Compared to the

Table 2: Sensitivity analysis for LCB confidence level δ on MBPP+. All δ values outperform the Pass-rate (m=5) baseline (52.9%), demonstrating robustness to this hyperparameter.

Method	Pass@1	Pass@5	Pass@10
Pass-rate (m=5)	52.9	–	–
LCB δ=0.01	**54.4**	72.8	**77.6**
LCB δ=0.05	53.9	72.4	76.7
LCB δ=0.10	54.0	72.0	76.3
LCB δ=0.20	54.0	72.2	76.8

standard Pass-rate (m=5) baseline, LCB improves Pass@1 by 3.5 percentage points on MBPP+ and 3.2 points on HumanEval+.

Notably, LCB (m=5) outperforms Pass-rate ($2m$=10) despite using only half the verifier compute (5 vs 10 test executions per rollout). This demonstrates that principled uncertainty quantification is more effective than simply executing more tests. The improvement transfers well from the training benchmark (MBPP+) to the held-out benchmark (HumanEval+), suggesting that LCB promotes more generalizable code generation.

4.3 Ablation Studies

Confidence Level Sensitivity. Table 2 shows that LCB performance is robust to the confidence level δ. All tested values ($\delta \in \{0.01, 0.05, 0.10, 0.20\}$) outperform the Pass-rate (m=5) baseline by 1.0–1.5 percentage points, with a spread of only 0.5 points across δ values. The most conservative setting (δ=0.01) achieves slightly higher Pass@1, but the differences are small. This indicates that δ is not a sensitive hyperparameter.

Prior Sensitivity. We also evaluated three Beta priors: Jeffreys (0.5, 0.5), Uniform (1, 1), and Informative (2, 2). All priors produce identical Spearman correlation (0.87) with oracle pass rates in pre-training analysis, indicating that the choice of prior has negligible effect when $m = 5$ tests are executed.

4.4 Training Dynamics Analysis

Figure 2 shows the reward-oracle correlation during training. All methods maintain high correlation ($>$0.92) throughout training, indicating no significant over-optimization. However, LCB shows the most stable trajectory with the smallest degradation from peak correlation (0.0004) compared to Pass-rate (m=5) (0.0017) and Pessimistic (0.0020). This suggests that the conservative LCB reward leads to more stable training dynamics.

4.5 Discussion

Mechanism Insight. Pre-training analysis reveals that LCB does not improve solution ranking over pass-rate—both achieve Spearman correlation of 0.87 with oracle pass rates, and within-group rankings are nearly identical (rank correlation 0.9996). Instead, LCB's benefit appears to come from reward value compression that affects GRPO's advantage normalization, leading to different gradient dynamics during training.

Limitations. Our experiments use different hyperparameters for LCB (5 epochs, lr $= 1 \times 10^{-5}$) compared to baselines (3 epochs, lr $= 5 \times 10^{-6}$), as the original LCB configuration with $\delta = 0.05$ caused reward dynamic range collapse. While we verified that LCB outperforms baselines under its optimized configuration, a fully controlled comparison would require additional hyperparameter search for all methods.

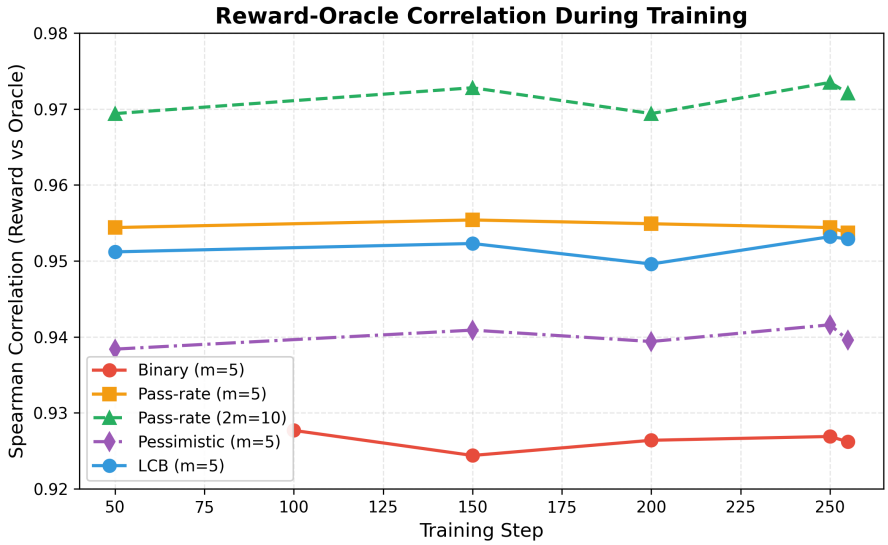

Figure 2: Reward-oracle Spearman correlation during GRPO training for all five reward functions. LCB (m=5) maintains stable correlation throughout training with the smallest degradation from peak (0.0004), while other methods show larger fluctuations.

5 Conclusion

We introduced the Lower Confidence Bound (LCB) reward for reinforcement learning from verifiable rewards in code generation. By modeling unit-test outcomes as Bernoulli trials and computing the δ-quantile of the Beta posterior, LCB provides a principled conservative estimate that accounts for finite-sample uncertainty inherent in executing only m tests per rollout. Experiments on MBPP+ and HumanEval+ demonstrate that LCB (m=5) achieves the best Pass@1 accuracy while using half the verifier compute of Pass-rate ($2m$=10), with improvements transferring to held-out benchmarks. The method is robust to hyperparameter choices and exhibits stable training dynamics. Limitations include the use of different hyperparameters across methods and incomplete understanding of the mechanism by which reward compression improves training. Future work may explore adaptive confidence levels and applications to other verification domains.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, H. Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program synthesis with large language models. *ArXiv*, abs/2108.07732, 2021.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé, Jared Kaplan, Harrison Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mo Bavarian, Clemens Winter, P. Tillet, F. Such, D. Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William H. Guss, Alex Nichol, Igor Babuschkin, S. Balaji, Shantanu Jain, A. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, M. Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, I. Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *ArXiv*, abs/2107.03374, 2021.

DeepSeek-AI, Daya Guo, Dejian Yang, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645:633 – 638, 2025.

Zhongmou He, Yee Man Choi, Kexun Zhang, Jiabao Ji, Junting Zhou, Dejia Xu, Ivan Bercovich, Aidan Zhang, and Lei Li. Hardtests: Synthesizing high-quality test cases for llm coding, 2025. URL https://arxiv.org/abs/2505.24098.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *ArXiv*, abs/2106.09685, 2021.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, An Yang, Rui Men, Fei Huang, Shanghaoran Quan, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-coder technical report. *ArXiv*, abs/2409.12186, 2024.

Aviral Kumar, Aurick Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. *ArXiv*, abs/2006.04779, 2020.

Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven C. H. Hoi. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. *ArXiv*, abs/2207.01780, 2022.

Jiate Liu, Yiqin Zhu, Kaiwen Xiao, Qiang Fu, Xiao Han, Wei Yang, and Deheng Ye. Rltf: Reinforcement learning from unit test feedback. *Trans. Mach. Learn. Res.*, 2023, 2023.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *ArXiv*, abs/2305.01210, 2024.

Zeyao Ma, Xiaokang Zhang, Jing Zhang, Jifan Yu, Sijia Luo, and Jie Tang. Dynamic scaling of unit tests for code reward modeling, 2025. URL https://arxiv.org/abs/2501.01054.

Zhihong Shao, Peiyi Wang, Qihao Zhu, R. Xu, Jun-Mei Song, Mingchuan Zhang, Y. K. Li, Yu Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *ArXiv*, abs/2402.03300, 2024.

Longwen Wang, Xuan'er Wu, Xiaohui Hu, Yirui Liu, Yuankai Fan, Kaidong Yu, Qizhen Weng, Wei Xi, and Xuelong Li. Verpo: Verifiable dense reward policy optimization for code generation. *ArXiv*, abs/2601.03525, 2026.

Yinjie Wang, Ling Yang, Ye Tian, Ke Shen, and Mengdi Wang. Co-evolving llm coder and unit tester via reinforcement learning. *ArXiv*, abs/2506.03136, 2025.

Yinglun Xu, Hangoo Kang, Tarun Suresh, Yuxuan Wan, and Gagandeep Singh. Learning a pessimistic reward model in rlhf, 2025a. URL https://arxiv.org/abs/2505.20556.

Zhenghao Xu, Qin Lu, Qingru Zhang, Liang Qiu, Ilgee Hong, Changlong Yu, Wenlin Yao, Yao Liu, Haoming Jiang, Lihong Li, Hyokun Yun, and Tuo Zhao. Ask a strong llm judge when your reward model is uncertain. *ArXiv*, abs/2510.20369, 2025b.

Bonan Zhang, Zhongqi Chen, Bowen Song, Qinya Li, Fan Wu, and Guihai Chen. Conf-clip: Confidence-weighted and clipped reward for reinforcement learning in llms. *ArXiv*, abs/2509.17730, 2025.