# GEN-RL: REWARDING REASONING THAT TRANSFERS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

In standard reinforcement learning with verifiable rewards (RLVR), a reasoning chain is rewarded for solving the particular problem that elicited it. But the reasoning we ultimately want models to learn should transfer: a broadly reusable strategy and an idiosyncratic solution should not be equally valuable. We introduce generalization-rl (*gen-rl*), which uses in-context learning as a direct signal for transfer. The model first produces a response to one problem, which is left in context as a demonstration for a set of subsequent problems. The first response is rewarded according to how well it enables downstream problem solving via in-context learning, while the downstream responses are themselves trained to make effective use of the demonstration and produce correct answers. We derive a principled policy-gradient estimator for learning this behavior end-to-end. We study transfer in one concrete form, easy-to-hard generalization: whether reasoning learned on problems of one difficulty extends to harder ones. Across multi-digit multiplication, knights & knaves, and standard competition math benchmarks, gen-rl outperforms standard RLVR both on problems as difficult as the training problems and on harder ones from held-out difficulty tiers. Gen-rl's improvement over standard RLVR is largest on the harder tiers, suggesting that it promotes reasoning strategies that extrapolate beyond the training distribution.

## 1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) trains a language model (LM) to reason by rewarding chains of thought that lead to successful outcomes. Yet, there is currently a mismatch between how standard RLVR rewards an LM and the reasoning we ultimately want the LM to learn: the standard process evaluates a reasoning chain based on whether it solves a particular problem, whereas our ultimate goal is to teach the model to discover general strategies that remain useful across many problems. The standard reward fails to distinguish an idiosyncratic solution from a general one. And, as LMs become more capable, a particular concern is that the standard process fails to select for strategies that will extrapolate to problems harder than those we generally have available for training. For standard RLVR, a correct solution is a correct solution, even if it relies on a mechanism whose utility ends at the edge of the training distribution.

One useful way to address these challenges is to curate diverse prompts and environments, and trust training scale and diversity to pay off on problems the model has yet to see. We explore a complementary approach: we explicitly design the training objective to encourage reasoning that transfers.

Our key idea is to use *in-context learning* (ICL; Brown et al., 2020) not as a test-time method of boosting performance, but instead as a training-time signal for whether a particular line of reasoning transfers. At its core, in-context learning involves extracting reusable structure from demonstrations and applying it to new instances. Therefore, a reasoning trace that contains a transferable strategy should help the model solve other problems when provided in context. Concretely, a rollout in our method takes the form

$$Q_1 \to A_1 \to \left\{ Q_2^{(m)} \to A_2^{(m)} \right\}_{m=1}^{M}, \tag{1}$$

where each of $A_1$ and $A_2$ stands for the model's entire response, including both its chain of thought and the final solution. After the model produces $A_1$ in response to an initial problem $Q_1$, we leave $(Q_1, A_1)$ in context as a demonstration and ask the model to solve a collection of $M$ subsequent "probe" problems. We use success on these downstream problems as the reward signal for $A_1$,



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

rather than the correctness of $A_1$ itself. We call this approach *gen-rl*. Because gen-rl builds on ICL — a capability that improves with model scale (Brown et al., 2020) — its training signal should only strengthen as models grow, which makes it complementary to the prevailing RLVR scaling paradigm.

Naively, gen-rl measures how well a given $A_1$ serves as an in-context demonstration. However, the process is more subtle. In Section 2.1, we show how to derive a principled objective for a gen-rl style rollout. A natural consequence of our principled objective is that the $A_2$ probe responses are also trained on, and also receive some advantage during the training process. That is, while the $A_1$ answers are never verified, the $A_2$ probe responses are verified and are rewarded for being correct during training. This arrangement forces the model to serve as both teacher and student. The policy must learn to generate reasoning chains that both serve as good demonstrations (as $A_1$), and that also maximize the probability of verifiably correct answers (as $A_2$). That is, the method does retain direct optimization pressure towards correctness. See Figure 1(a).

To evaluate if gen-rl enables generalization to harder problems, we need clean testbeds in which "harder than training" is well defined. We start with multi-digit multiplication and knights & knaves logic puzzles, each of which admits a parametric difficulty adjustment. For multiplication, we can make problems harder by increasing the number of digits, and for knights & knaves we can increase the number of "inhabitants" to whom the model must assign a label as a "knight" (truthteller) or "knave" (liar) (Figure 3). For a more realistic task, we then move on to competition mathematics. For competition math, we band problems by the base model's own solve rate, measured empirically. In all of our tasks, we train on a well-defined range of difficulty, and evaluate both on held-out problems within this range, and on harder problems. We evaluate in the typical way, on single-question accuracy with no demonstration in context.

In each of our testbeds gen-rl outperforms a standard RLVR control on every tier, and its advantage, measured against RLVR's own gain over the base model, grows with difficulty and peaks above the training range (Figure 1(b)). On multiplication the margin rises from $+0.9$ points on the easiest tier to $+12.0$ on the hardest; on knights & knaves puzzles with twice as many inhabitants as any training puzzle, the base model solves 0%, standard RLVR solves 2.1%, and gen-rl reaches 12.6%; and on the harder math bands gen-rl delivers more than 1.4 times RLVR's gain.

Our results suggest a broader paradigm for encouraging generalizable reasoning with RL: we can redesign the training task so that — while still training end-to-end from a verifiable reward signal, with no subjective external rubrics — general strategies emerge bottom-up from transfer itself.

## 2 GEN-RL

The objective in gen-rl is the model's success on downstream problems when its own response to an earlier problem is left in context as a demonstration. We first formalize this concept as a training objective (Section 2.1), and then derive the group-relative advantages used to estimate its gradient (Section 2.2).

### 2.1 OBJECTIVE

Recall the form of a gen-rl rollout from (1). A question $Q_1$ is drawn from a distribution $\mathcal{D}$. The policy $\pi_\theta$ produces a response $A_1 \sim \pi_\theta(\cdot \mid Q_1)$, which we call the *prefix*, since it remains in context for everything that follows. We then draw $M$ probe questions $Q_2^1, \ldots, Q_2^M$ independently from a probe distribution $\mathcal{D}'$, and for each probe the policy produces a probe response, conditioned on $Q_1$ and $A_1$: $A_2^m \sim \pi_\theta(\cdot \mid Q_1, A_1, Q_2^m)$. The probe distribution may depend on $Q_1$, a dependence we suppress in the notation: in our experiments, probes are drawn from a harder difficulty tier than $Q_1$ wherever one is available (Section 3.1). A verifier scores each probe response, $c(A_2^m) \in \{0, 1\}$; the correctness of $A_1$'s own answer to $Q_1$ never enters the reward.

A natural approach to reinforcement learning on a rollout with this structure would be to define some reward over the probe set, and seek to maximize that reward. One natural reward is just the success rate over the probes: $R(\{A_2^m\}) = \frac{1}{M}\sum_{m=1}^{M} c(A_2^m)$. We choose instead a monotone, concave

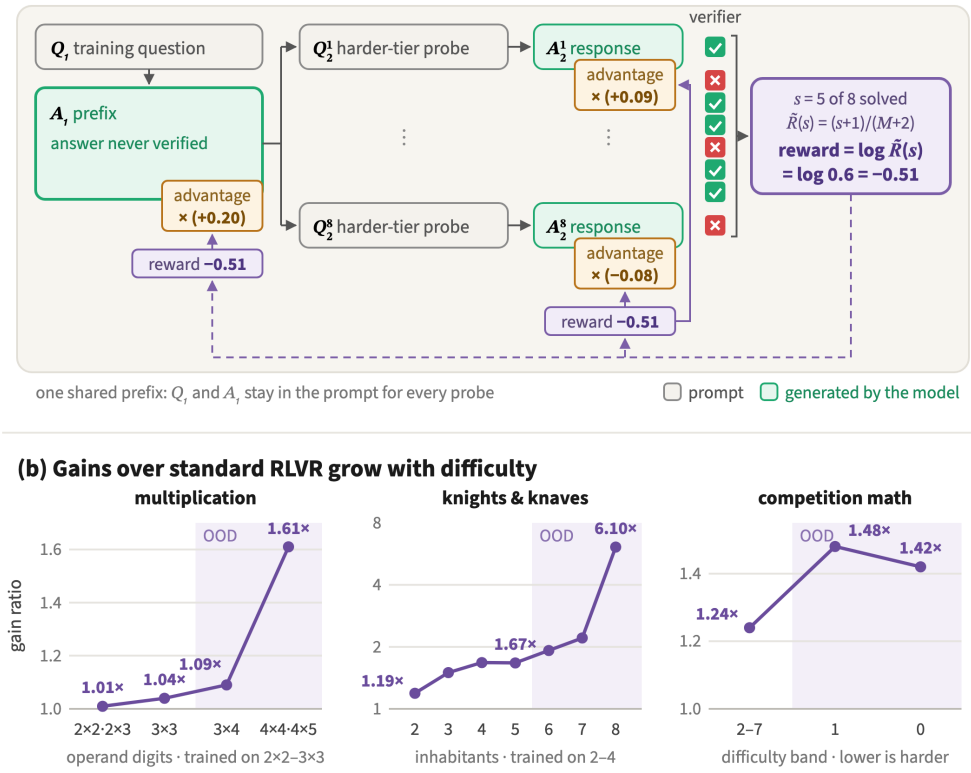



Figure 1: **gen-rl rewards reasoning that transfers. (a)** A two-turn rollout. The policy answers a training question $Q_1$ with an answer $A_1$. Both $Q_1$ and $A_1$ stay in context as a demonstration while the same policy answers each of $M$ probe questions from harder tiers. Only the probe responses are verified. The number of probes solved, $s$, provides the basis for the reward $\log \widetilde{R}(s)$. $A_1$'s reward acts as an empirical measure of how well it functions as an in-context demonstration. $A_2$ probe responses also receive reward, maintaining gradient pressure for correctness. Section 2.2 below describes how to calculate group relative advantages for each part of the tree structure pictured above. **(b)** Our results show that gen-rl improves over standard RLVR, especially on problems that are more difficult than the training distribution: gain ratio, (gen-rl − base) / (control − base), across difficulty tiers of three tasks; shaded tiers are harder than anything seen in training.

transform of this simple success rate — the log of a smoothed average:

$$\log \widetilde{R}(s), \qquad \text{where} \quad \widetilde{R}(s) := \frac{s+\alpha}{M+2\alpha} \quad \text{and} \quad s := \sum_{m=1}^{M} c(A_2^m).$$

Here, $s$ is the number of probes solved and $\alpha > 0$ is a smoothing constant; we use $\alpha = 1$. The smoothing is necessary because the success count is 0 whenever every probe fails, and $\log 0$ is undefined. The smoothing keeps the log reward finite for every rollout while leaving the ranking of rollouts unchanged.

In Appendix A, we motivate this choice of reward in greater detail. In particular, we show how this form of the reward relates to viewing gen-rl as a conditional latent variable model that captures the in-context learning effect of $Q_1$ and $A_1$ on $A_2$. More intuitively, the log smoothed success rate changes what one more solved probe is worth (Figure 2). On a hard probe set (meaning one in which

the success counts are quite low across rollouts), an $A_1$ that leads to one more probe success than its peers in the same group is rewarded much more for that additional success than it would be on an easy probe set, where most probes are already solved. When the goal is transfer to harder problems, this may be the emphasis we want: the demonstrations that gain some traction on hard probe sets, where success is still rare, shape the update more than further gains on problems on easier sets. Consistent with this, in our ablation the model trained with the log version of the reward matches a model trained with the linear reward on trained difficulty tiers, but does substantially better when evaluated on held-out harder tiers (Appendix A, Table 5). We do not study under what circumstances using the linear reward $R$ (or some other reward over the set) might be preferable, and we use the $\log \widetilde{R}(s)$ version for our principal experiments.

Maximizing a reward defined in this way gives the following objective:

$$J_{\text{gen}}(\theta) \;:=\; \mathbb{E}_{A_1\sim\pi_\theta(\cdot|Q_1)}\,\mathbb{E}_{\{Q_2^m\}\sim\mathcal{D}'}\,\mathbb{E}_{\{A_2^m\}\sim\pi_\theta(\cdot|Q_1,A_1,Q_2^m)}\big[\log \widetilde{R}(s)\,\big]. \tag{2}$$

The gradient takes the usual score-function (REINFORCE; Williams, 1992) form, in which the reward multiplies the log-probability gradients of both turns:

$$\begin{aligned}\nabla_\theta J_{\text{gen}} = \mathbb{E}_{A_1\sim\pi_\theta(\cdot|Q_1)}\,\mathbb{E}_{\{Q_2^m\}\sim\mathcal{D}'}\,\mathbb{E}_{\{A_2^m\}\sim\pi_\theta(\cdot|Q_1,A_1,Q_2^m)}\\ \Big[\log \widetilde{R}(s)\,\Big(\nabla_\theta \log \pi_\theta(A_1 \mid Q_1) + \textstyle\sum_m \nabla_\theta \log \pi_\theta(A_2^m \mid Q_1, A_1, Q_2^m)\Big)\Big].\end{aligned} \tag{3}$$

That is, both the outer traces (the $A_1$ prefixes) and the inner traces (the $A_2$ probe responses) receive some reward, leading to the teacher-student dynamic described in the introduction.

### 2.2 Group-relative advantages

**Binary matrix view of a group rollout.** We estimate the gradient derived above with a group-relative policy-gradient estimator in the style of GRPO (Shao et al., 2024). For each $Q_1$, we roll out $G$ attempts by the model at that same prompt. We index those rollouts by $i$, writing $A_1^i$ for the $i$th prefix attempt. Each of those $G$ rollouts gets the same set of $M$ probes. We index the probes by $m$, such that $A_2^{i,m}$ is the model's response to probe $Q_2^m$ with $A_1^i$ as the demonstration.

We also need notation for the verifier bit on each probe response: let $c_i^m := c(A_2^{i,m})$.

Using this notation, a convenient way to view the outcome of a single group rollout in the style of (1) is as a $G$ by $M$ binary matrix:

$$\begin{array}{cccc} & Q_2^1 & \cdots & Q_2^M & \\ A_1^1 & c_1^1 & \cdots & c_1^M & s_1 = \sum_m c_1^m \\ \vdots & \vdots & & \vdots & \vdots \\ A_1^G & c_G^1 & \cdots & c_G^M & s_G = \sum_m c_G^m \end{array}$$

The row sums in this matrix give the success counts, $s_i$, for calculating $\log \widetilde{R}(s_i)$. The column direction feeds into the baselines for the probe responses, as described below.

**Prefix advantages.** For the prefixes, the natural baseline for a GRPO-style advantage is the leave-one-out mean of the other rows' rewards (Kool et al., 2019; Ahmadian et al., 2024),

$$\mathcal{A}_i^1 \;=\; \log \widetilde{R}(s_i) \;-\; \frac{1}{G-1}\sum_{j\neq i} \log \widetilde{R}(s_j). \tag{4}$$

That is, the log smoothed success rate gives a kind of "transfer score" for how well a given $A_1^i$ serves as an in-context demonstration. We center these scores at the (leave-one-out) group mean, in the standard way. Prefixes whose transfer score is higher than that of their peers in the group (prefixes that have shown themselves to be relatively strong ICL demonstrations) receive positive advantage, and those whose score is lower (relatively weak ICL demonstrations) receive negative advantage.

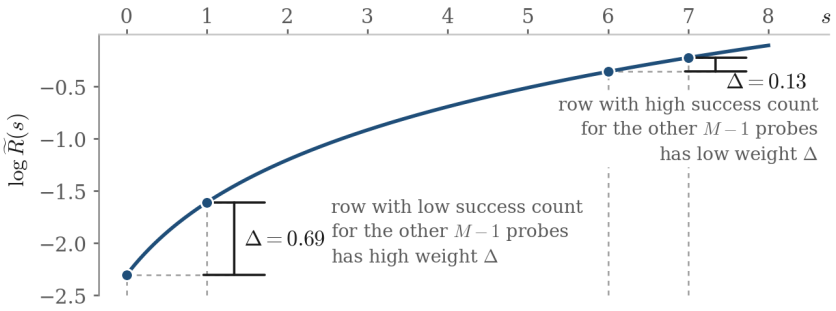



Figure 2: The weight $\Delta_i^m$ on a probe response is the jump in the row's smoothed log success rate $\log \widetilde{R}(s)$ from one more solved probe, evaluated at the row's count over its other probes (shown here with $M = 8$, $\alpha = 1$).

**Probe advantages.** For the probe responses, the natural baseline seems less obvious, because the reward $\log \widetilde{R}(s_i)$ depends on the whole row rather than on any single probe response. Consider the part of the gradient we are trying to estimate for the probe responses:

$$\nabla_\theta^{\text{probe}} J_{\text{gen}} = \mathbb{E}\Big[\log \widetilde{R}(s_i) \sum_{m=1}^{M} \nabla_\theta \log \pi_\theta(A_2^{i,m} \mid Q_1, A_1^i, Q_2^m)\Big]$$
$$= \sum_{m=1}^{M} \mathbb{E}\Big[\log \widetilde{R}(s_i)\, \nabla_\theta \log \pi_\theta(A_2^{i,m} \mid Q_1, A_1^i, Q_2^m)\Big]. \tag{5}$$

Recall that $s_i = \sum_m c_i^m$ is the success count for the $i$th row, and define $s_i^{-m} := s_i - c_i^m$, the same count leaving out probe $m$ itself. For a given fixed $m$, we can now rewrite the reward as an affine function of that probe's own correctness bit:

$$\log \widetilde{R}(s_i) \;=\; \log \widetilde{R}(s_i^{-m}) \;+\; c_i^m\, \Delta_i^m, \qquad \Delta_i^m \;:=\; \log \widetilde{R}(s_i^{-m}+1) - \log \widetilde{R}(s_i^{-m}).$$

That is, $\Delta_i^m$ is a scalar weight that captures the effect of the log: it represents the increase in the row's reward from one more solved probe, evaluated with probe $m$ left out. Figure 2 shows how this weight scales with the log depending on the success count for a given row.

In Appendix B we demonstrate that the gradient in (5) can be rewritten in terms of $\Delta_i^m$ as follows:

$$\nabla_\theta^{\text{probe}} J_{\text{gen}} = \sum_{m=1}^{M} \mathbb{E}\Big[c_i^m\, \Delta_i^m\, \nabla_\theta \log \pi_\theta(A_2^{i,m} \mid Q_1, A_1^i, Q_2^m)\Big]$$
$$= \sum_{m=1}^{M} \mathbb{E}\Big[\big(c_i^m - b_{i,m}\big)\, \Delta_i^m\, \nabla_\theta \log \pi_\theta(A_2^{i,m} \mid Q_1, A_1^i, Q_2^m)\Big]$$
$$= \sum_{m=1}^{M} \mathbb{E}\Big[\Big(c_i^m - \frac{1}{G-1}\sum_{j\neq i} c_j^m\Big)\, \Delta_i^m\, \nabla_\theta \log \pi_\theta(A_2^{i,m} \mid Q_1, A_1^i, Q_2^m)\Big].$$

The second equality holds for any baseline $b_{i,m}$ that does not depend on $A_2^{i,m}$. The third equality chooses one such baseline: because all of the $G$ first-turn rollouts are tested with the same collection of $M$ probes, we can center each correctness bit $c_i^m$ on the model's other attempts at the same $m^{th}$ question. This choice of baseline gives the following advantage calculation:

$$\mathcal{A}_{i,m}^2 \;=\; \Big(c_i^m \;-\; \frac{1}{G-1}\sum_{j\neq i} c_j^m\Big)\cdot \Delta_i^m. \tag{6}$$

That is, the advantage is calculated relative to a leave-one-out column average, then weighted by $\Delta_i^m$.

**(a) Multi-digit multiplication**

```
What is 47 * 83?          (2×2)
What is 4817 * 39265?     (4×5)
```

| digits in the two factors | | | |
|---|---|---|---|
| $2{\times}2 \cdot 2{\times}3$ | $3{\times}3$ | $3{\times}4$ | $4{\times}4 \cdot 4{\times}5$ |
| trained | | held out (OOD) | |

**(b) Knights & knaves**

Knights always tell the truth, and knaves always lie. You meet 3 inhabitants: Chloe, Mason, and Aria. "Aria is a knave," Chloe declared. Mason said, "Aria is a knave." "Chloe is a knight or Aria is a knight," Aria mentioned. So who is a knight and who is a knave?

| inhabitants | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| candidate assignments | 4 | 8 | 16 | 32 | 64 | 128 | 256 |
| | trained | | | held out (OOD) | | | |

Figure 3: **Tasks with parametric difficulty adjustment.** One parameter sets the difficulty: the number of digits, or the number of inhabitants. Unshaded tiers are used for training and for in-distribution evaluation on fresh problems; shaded tiers are harder than anything seen in training and are used for evaluation only.

**Update.** With the advantages above, we can now give the estimated gradient for one group as:

$$\widehat{\nabla}_\theta J_{\text{gen}} \;=\; \frac{1}{G}\sum_{i=1}^{G}\Big[\,\mathcal{A}_i^1\,\nabla_\theta \log \pi_\theta(A_1^i \mid Q_1) \;+\; \sum_{m=1}^{M} \mathcal{A}_{i,m}^2\,\nabla_\theta \log \pi_\theta(A_2^{i,m} \mid Q_1, A_1^i, Q_2^m)\Big].$$

In practice, we add a small format term to each advantage (Appendix C). We apply the update through the clipped surrogate of PPO (Schulman et al., 2017), with no KL penalty. We train on every response in the group, prefix and probe alike.

## 3 Experiments

Our experiments ask three questions. Does gen-rl improve on standard RLVR when evaluated in the typical setting, on single questions with no demonstration in context? Does its advantage grow on problems harder than any seen in training, as the method intends? And do the gains hold on a more realistic task, under compute matching, and on external benchmarks? The answer to each is yes.

### 3.1 Tasks and setup

**Tasks and difficulty tiers.** We call multi-digit multiplication and knights & knaves (Stojanovski et al., 2025) the *parametric tasks*, because a single parameter sets their difficulty: the number of digits in the operands, or the number of inhabitants in the puzzle (Figure 3). For competition mathematics we use the AceReason-Nemotron math training set (Chen et al., 2025), and band each problem by how many of 8 attempts the base model solves, so that band 0, never solved, is the hardest. We train on bands 2–7. Band 1 ("near-OOD") and band 0 ("far-OOD") are held out for evaluation, as is a *retention* set from band 8, which the base model always solves and which we drop from training as too easy.

We use *in-distribution* (ID) and *out-of-distribution* (OOD) in the special sense of difficulty ratings. ID problems are fresh, held-out problems at the levels of difficulty the model encountered during training; OOD problems are harder than any problem the model has seen during training, whether as $Q_1$ or $Q_2$. During training, we sample probes from a harder difficulty tier than the $Q_1$ that they follow, or from the same tier when $Q_1$ is already in the hardest tier of the training distribution. All evaluation is done at the same token generation caps as training.

**Models, training, and controls.** For the parametric tasks we train Qwen2.5-1.5B-Instruct with LoRA, at $G{=}12$ rollouts per prompt. For competition math we train Qwen3-1.7B in "non-thinking" mode (that is, the model still generates a chain of thought, but is not in its "reasoning model" mode) with full finetuning, at $G{=}8$. All experiments use $M{=}8$ probes per $Q_1$ for the method arm, and every run trains for 500 steps. Rewards are strictly binary, with 1 awarded solely for fully, verifiably correct answers, plus a small format reward (Appendix C).

Table 1: **Multiplication results.** Results shown as of the final checkpoint of training (step 500). Section 3.2 defines "gain ratio." "$\Delta$ (pp)" is gen-rl correctness $-$ control correctness. Intervals are 95% paired bootstraps over items with 10,000 resamples; a star marks an interval that excludes 0 for $\Delta$ or 1 for the ratio. The gain-ratio column is shaded on a log scale.

| | correctness (%) | | | | |
|---|---|---|---|---|---|
| tier | base | control | **gen-rl** | $\Delta$ (pp) | gain ratio |
| $2\times2 \cdot 2\times3$ (ID) | 10.2 | 96.4 | 97.3 | $+0.9$ | 1.01 |
| $3\times3$ (ID) | 4.5 | 90.3 | 93.9 | $+3.6^*$ | $1.04^*$ |
| $3\times4$ (OOD+1) | 0.6 | 72.5 | 79.3 | $+6.8^*$ | $1.09^*$ |
| $4\times4 \cdot 4\times5$ (OOD+2) | 0.1 | 19.9 | 31.9 | $+12.0^*$ | $1.61^*$ |

Table 2: **Knights & knaves results.** Conventions as in Table 1.

| | correctness (%) | | | | |
|---|---|---|---|---|---|
| tier | base | control | **gen-rl** | $\Delta$ (pp) | gain ratio |
| 2 inhabitants (ID) | 0.6 | 53.5 | 63.5 | $+9.9^*$ | $1.19^*$ |
| 3 inhabitants (ID) | 0.3 | 33.9 | 50.7 | $+16.8^*$ | $1.50^*$ |
| 4 inhabitants (ID) | 0.0 | 23.7 | 39.8 | $+16.1^*$ | $1.68^*$ |
| 5 inhabitants (OOD+1) | 0.0 | 16.4 | 27.3 | $+10.9^*$ | $1.67^*$ |
| 6 inhabitants (OOD+2) | 0.1 | 11.2 | 21.4 | $+10.2^*$ | $1.92^*$ |
| 7 inhabitants (OOD+3) | 0.0 | 6.5 | 14.5 | $+7.9^*$ | $2.21^*$ |
| 8 inhabitants (OOD+4) | 0.0 | 2.1 | 12.6 | $+10.5^*$ | $6.10^*$ |

The control is standard RLVR: GRPO (Shao et al., 2024) on the same training data, in the usual one-question-at-a-time fashion, at the same $G$ as the method. For competition math we add a *compute-matched* control with $G$=72 rollouts per prompt, which is the number of sequences gen-rl trains on per prompt (its 8 prefixes and their $8 \times 8$ probe responses); it used slightly more GPU-hours than gen-rl.[1] Appendix D gives the remaining details of our training regime and hyperparameters.

### 3.2 Main results

**Gen-rl outperforms standard RLVR on all three tasks.** On the parametric tasks, gen-rl is ahead of the control on all 11 difficulty tiers, ID and OOD alike (Tables 1 and 2). The gain is significant on every tier but the easiest tier of multiplication, where both arms are already above 96%. On competition math, gen-rl is significantly ahead on the ID holdout and on both OOD bands (Table 3). These gaps are not an artifact of the final checkpoint: they emerge during training and persist to the end (Appendix E).

**The advantage grows with difficulty.** Measured against what standard RLVR itself achieves, gen-rl's advantage is smallest on the easiest problems and largest on the hardest, which lie beyond the training distribution (Figure 1(b)). To compare tiers that start from different base-model performance and offer different amounts of headroom, we report the *gain ratio*, (gen-rl $-$ base) / (control $-$ base): gen-rl's improvement over the base model, in units of the standard control's own improvement.[2] The gain ratio rises almost monotonically with difficulty on both parametric tasks, from 1.01 to 1.61 on multiplication and from 1.19 to 6.10 on knights & knaves. On the hardest multiplication tier, gen-rl takes the control's 19.9% to 31.9%. On 8-inhabitant puzzles, twice the size of any seen in training, the base model solves 0 of 1,500 attempts and the control 2.1%; gen-rl solves 12.6%.

**The pattern holds on competition math, and survives compute matching.** On competition math, gen-rl improves on the standard control by 3.6 points on the ID holdout, 7.9 on the near-OOD band and 4.8 on the far-OOD band, and the gain ratios on the two OOD bands (1.48 and 1.42)

[1] 427.0 A100-hours versus 414.6.

[2] The gain ratio is undefined when the control's own gain over the base model is not significantly positive; in those cells we report the raw evaluation statistics only.

Table 3: **Competition math results.** The $G{=}72$ control is the compute-matched control of Section 3.1. The gain ratio is taken against the $G{=}8$ control (which outperformed the compute-matched control); — marks a leg on which that control's gain over the base model is not significantly positive. Other conventions as in Table 1.

| | correctness (%) | | | | $\Delta$ (pp), gen-rl $-$ control | | gain ratio |
|---|---|---|---|---|---|---|---|
| leg | base | control $G{=}8$ | control $G{=}72$ | **gen-rl** | vs control $G{=}8$ | vs control $G{=}72$ | vs control $G{=}8$ |
| ID holdout (bands 2–7) | 55.0 | 70.3 | 62.8 | 73.9 | $+3.6^*$ | $+11.1^*$ | $1.24^*$ |
| near-OOD (band 1) | 14.1 | 30.6 | 24.4 | 38.5 | $+7.9^*$ | $+14.2^*$ | $1.48^*$ |
| far-OOD (band 0) | 4.5 | 16.0 | 13.2 | 20.8 | $+4.8^*$ | $+7.5^*$ | $1.42^*$ |
| retention (band 8) | 94.5 | 94.2 | 92.8 | 95.3 | $+1.2$ | $+2.5$ | — |

Table 4: **External competition math benchmarks.** AIME 2024 and 2025 are raw benchmark sets; the POLARIS legs apply the same base-model difficulty measurement as AceReason to a different source, so each row matches a band of Table 3. Conventions as in Table 3.

| | correctness (%) | | | | $\Delta$ (pp), gen-rl $-$ control | | gain ratio |
|---|---|---|---|---|---|---|---|
| leg | base | control $G{=}8$ | control $G{=}72$ | **gen-rl** | vs control $G{=}8$ | vs control $G{=}72$ | vs control $G{=}8$ |
| AIME 2024 | 12.5 | 17.5 | 10.4 | 23.1 | $+5.6^*$ | $+12.7^*$ | — |
| AIME 2025 | 10.4 | 19.2 | 12.5 | 18.3 | $-0.8$ | $+5.8^*$ | 0.90 |
| POLARIS near-OOD (band 1) | 13.8 | 21.8 | 18.6 | 28.3 | $+6.5^*$ | $+9.7^*$ | $1.82^*$ |
| POLARIS far-OOD (band 0) | 3.0 | 7.0 | 6.5 | 9.2 | $+2.2^*$ | $+2.8^*$ | $1.55^*$ |
| POLARIS retention (band 8) | 92.2 | 86.3 | 81.8 | 91.8 | $+5.5^*$ | $+10.0^*$ | — |

again exceed the ratio on the ID holdout (1.24). Against the compute-matched control, its margins on the same three legs are 11.1, 14.2 and 7.5 points. None of this comes at the expense of easy problems: on the retention set there is no significant difference between the arms.

**The gains transfer to external benchmarks.** The competition-math models were trained only on AceReason problems; Table 4 evaluates them elsewhere. On AIME 2024, gen-rl scores 23.1% to the control's 17.5%, while on AIME 2025 the two arms are indistinguishable, each gaining roughly 8 points over the base model. On POLARIS, a separate competition-math source that we band with the same base-model measurement, gen-rl is ahead on both OOD bands, with gain ratios (1.82 and 1.55) that exceed those on AceReason itself. On the POLARIS retention band the control loses ground relative to the base model (92.2% to 86.3%), and gen-rl does not (91.8%).

## 4 Related work

**In-context learning and in-context teaching.** A striking capability of modern language models is in-context learning: models can adapt their behavior from examples provided in the prompt, without any parameter updates (Brown et al., 2020; Wei et al., 2022). These results suggest that a demonstration can communicate not only an answer, but reusable structure that changes how the model solves subsequent problems. Recent work has studied this phenomenon through the lens of *teaching*: Zhao et al. (2024) use model-generated rationales as demonstrations to improve reasoning on unseen questions, and work on *learning to teach* optimizes a teacher from feedback on the learning of a student (Fan et al., 2018; Ning et al., 2024). Gen-rl transforms ICL utility from an inference-time phenomenon into the training objective itself: rather than assuming that the sole measure of a good reasoning trace is solving the problem that elicited it, gen-rl also asks whether traces teach the model something useful for what comes next. Mei et al. (2026) also observe that better reasoning makes better demonstrations and use ICL to measure a trace's demonstration utility; in their In-Context RLVR, however, fixed reference demonstrations from a stronger model are prepended to training questions and each rollout is still rewarded for its own answer, so that demonstration utility enters only as an implicit reweighting of the standard reward.

**Meta-learning.** Gen-rl is also connected to meta-learning, where learning systems are optimized for performance *after* some adaptation process (Finn et al., 2017; Duan et al., 2016). ICL can be viewed as a form of fast adaptation in which previous examples change subsequent computation without changing the model parameters; under this perspective, the first-turn rollout induces an adaptation through context, and the gen-rl objective trains the model to perform well after that adaptation. Sleep-time compute (Lin et al., 2025) likewise asks whether computation performed now is useful for questions that arrive later, but studies how to allocate inference computation given a context, whereas gen-rl trains the content of that computation end-to-end.

**Alternative objectives for reasoning.** A growing line of work treats chain-of-thought as latent computation whose value is determined through an observable downstream outcome (Zelikman et al., 2022; Phan et al., 2023; Zelikman et al., 2024; Ruan et al., 2025; Gandhi et al., 2026); these works informed our interpretation of gen-rl as a conditional latent variable model (Appendix A). Our $\log \widetilde{R}(s)$ reward also connects to Davis & Recht (2025), which shows that common binary-reward RL algorithms optimize different monotone transformations of the probability of correctness, and Tajwar et al. (2026), which develops sample-based objectives that interpolate between expected reward and maximum likelihood over successful trajectories.

**Credit assignment and set-valued reinforcement learning.** Finally, gen-rl is related to methods that branch a language-model trajectory to obtain richer credit-assignment signals. VinePPO (Kazemnejad et al., 2025) samples continuations from intermediate reasoning states to estimate the value of earlier steps for success later in the *same* problem; gen-rl instead assigns credit across problems. Set reinforcement learning optimizes objectives defined jointly over multiple generations (Orney et al., 2026; Hamid et al., 2026). Gen-rl's reward is likewise defined over a set: a single scalar computed from the outcomes over the $Q_2$ probes.

## 5 Discussion

**Limitations: compute.** Gen-rl spends more compute per training question than standard RLVR, because every first-turn response is followed by $M$ probe responses. Its advantage on competition math survives a compute-matched control (Section 3.2), but we have made no attempt to make the method cheap. We fixed $M{=}8$ and never swept it, so a much smaller probe set may work as well. Nor did we explore applying gen-rl on only some training steps, or only late in training, or at variable $M$ at different stages of training. We did not attempt to capitalize on the $(Q_1, A_1)$ context that a prefix's $M$ probe responses share to speed up the policy update. Our aim here is to introduce the idea and a principled objective for it; mapping the trade-off between probe compute and transfer is an important next step.

**Promising future directions: non-verifiable $Q_1$; training verifiers; related $Q_1$/$Q_2$.** Because $A_1$ is never verified, nothing in the objective requires $Q_1$ to be verifiable. Gen-rl could therefore train responses to questions with no checkable answer, through their effect on later questions that do have one, which would extend RLVR beyond verifiable domains. A related case is training models to act as verifiers — a binary verdict is often right by luck, so outcome reward in the verification setting reinforces many unsound reasoning chains, whereas under gen-rl a lucky verdict (as $A_1$) earns nothing unless its reasoning helps elsewhere.

Another direction is to pair each $Q_1$ with a topically related set of $Q_2$ probes. For example, in competition math, if $Q_1$ is a combinatorics question, the probe set could be sampled from other combinatorics questions. This setup would tighten the connection to meta-learning.

**Rewarding transfer.** Standard RLVR leans on the breadth of its training problems and environments to favor general strategies over local ones. All negotiation of the relationship among problems, even among problems from the same testbed, occurs only in weights. But the way that attempts to solve related questions interact with each other — help one another, or hinder one another — may itself be a rich source for a model's learning (cf. Pres et al., 2026; Wen et al., 2025). We hope that gen-rl contributes to a broader approach in which the relationships among solutions become an explicit part of the training objective.

### AI USE STATEMENT

In this work, we used generative AI tools extensively, including to: *help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, clean and reformat datasets, support qualitative and thematic data analysis, and interpret results*. We have not used generative AI tools for *generating synthetic datasets*, and *assisting with translation* is not applicable to this work. Additionally, we used generative AI tools to: *create or modify scientific figures or images, suggest experimental parameters, create or edit software code, creation of artifacts, draft parts of a research paper, summarize or analyse existing literature, discover research topics or identify gaps, brainstorming, sourcing/searching for information, edit a research paper to improve readability, identify relevant literature, format references, suggest a structure for a research paper, propose a title or keywords for a research paper*. We have reviewed all AI-assisted work. Every AI-drafted passage was read by at least one author, and often revised extensively. Every number in the tables and figures is generated by script from the raw evaluation outputs, and a regression check confirms cell for cell that the typeset tables agree with the consolidated results; the configuration tables in Appendix D are generated from the training configurations. The code has a test suite. The derivations in Appendices A and B were checked line by line by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

### REPRODUCIBILITY STATEMENT

The objective and its estimator are specified in Section 2; the log transform is motivated in Appendix A and the probe advantage derived in Appendix B. Appendix D collects the experimental details: the settings of every run (Tables 6 and 7), the prompts and rollout format, the construction of the training data and probes, the evaluation protocol with the items and attempts behind each reported cell (Table 9), and the software. All base models are publicly available, as are the knights & knaves task from Reasoning Gym (Stojanovski et al., 2025) and the competition math datasets: the AceReason-Nemotron math set (Chen et al., 2025), POLARIS (An et al., 2025), and AIME 2024 and 2025. Our implementation of multi-digit multiplication is custom, but described in detail in Figure 3 and Appendix D.4. The statistical conventions of the tables are given in the caption of Table 1, and Appendix E shows the evaluation curves over training.

## REFERENCES

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. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

Chenxin An, Zhihui Xie, Xiaonan Li, Lei Li, Jun Zhang, Shansan Gong, Ming Zhong, Jingjing Xu, Xipeng Qiu, Mingxuan Wang, and Lingpeng Kong. POLARIS: A post-training recipe for scaling reinforcement learning on advanced reasoning models. `https://hkunlp.github.io/blog/2025/Polaris`, 2025.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL `https://arxiv.org/abs/2005.14165`.

Shiyi Cao, Sumanth Hegde, Dacheng Li, Tyler Griggs, Shu Liu, Eric Tang, Jiayi Pan, Xingyao Wang, Akshay Malik, Graham Neubig, Kourosh Hakhamaneshi, Richard Liaw, Philipp Moritz, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. SkyRL-v0: Train real-world long-horizon agents via reinforcement learning. `https://github.com/NovaSky-AI/SkyRL`, 2025.

Yang Chen, Zhuolin Yang, Zihan Liu, Chankyu Lee, Peng Xu, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. AceReason-Nemotron: Advancing math and code reasoning through reinforcement learning, 2025. URL https://arxiv.org/abs/2505.16400.

Damek Davis and Benjamin Recht. What is the objective of reasoning with reinforcement learning?, 2025. URL https://arxiv.org/abs/2510.13651.

Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. $RL^2$: Fast reinforcement learning via slow reinforcement learning, 2016. URL https://arxiv.org/abs/1611.02779.

Yang Fan, Fei Tian, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. Learning to teach, 2018. URL https://arxiv.org/abs/1805.03643.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017. URL https://arxiv.org/abs/1703.03400.

Kanishk Gandhi, Agam Bhatia, and Noah D. Goodman. Learning to simulate human dialogue, 2026. URL https://arxiv.org/abs/2601.04436.

Jubayer Ibn Hamid, Ifdita Hasan Orney, Michael Y. Li, Omar Shaikh, Yoonho Lee, Dorsa Sadigh, Chelsea Finn, and Noah Goodman. SPIRAL: Learning to search and aggregate, 2026. URL https://arxiv.org/abs/2606.23595.

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. In *International Conference on Learning Representations (ICLR)*, 2022.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Refining credit assignment in RL training of LLMs, 2025. URL https://arxiv.org/abs/2410.01679.

Wouter Kool, Herke van Hoof, and Max Welling. Buy 4 REINFORCE samples, get a baseline for free! In *ICLR Workshop on Deep Reinforcement Learning Meets Structured Prediction*, 2019.

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

Kevin Lin, Charlie Snell, Yu Wang, Charles Packer, Sarah Wooders, Ion Stoica, and Joseph E. Gonzalez. Sleep-time compute: Beyond inference scaling at test-time, 2025. URL https://arxiv.org/abs/2504.13171.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-like training: A critical perspective, 2025. URL https://arxiv.org/abs/2503.20783.

Tiehua Mei, Minxuan Lv, Leiyu Pan, Zhenpeng Su, Hongru Hou, Hengrui Chen, Ao Xu, and Deqing Yang. Good reasoning makes good demonstrations: Implicit reasoning quality supervision via in-context reinforcement learning. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026. arXiv:2603.09803.

Xuefei Ning, Zifu Wang, Shiyao Li, Zinan Lin, Peiran Yao, Tianyu Fu, Matthew B. Blaschko, Guohao Dai, Huazhong Yang, and Yu Wang. Can LLMs learn by teaching for better reasoning? A preliminary study, 2024. URL https://arxiv.org/abs/2406.14629.

Ifdita Hasan Orney, Jubayer Ibn Hamid, Shreya S Ramanujam, Shirley Wu, Hengyuan Hu, Noah Goodman, Dorsa Sadigh, and Chelsea Finn. Poly-EPO: Training exploratory reasoning models, 2026. URL https://arxiv.org/abs/2604.17654.

Du Phan, Matthew Douglas Hoffman, David Dohan, Sholto Douglas, Tuan Anh Le, Aaron Parisi, Pavel Sountsov, Charles A. Sutton, Sharad Vikram, and Rif A. Saurous. Training chain-of-thought via latent-variable inference. In *Advances in Neural Information Processing Systems*, 2023.

Itamar Pres, Belinda Z. Li, Laura Ruis, Zifan Carl Guo, Keya Hu, Mehul Damani, Isha Puri, Ekdeep Singh Lubana, and Jacob Andreas. Position: It's time to optimize LLMs for self-consistency, 2026. URL https://arxiv.org/abs/2608.05188.

Yangjun Ruan, Neil Band, Chris J. Maddison, and Tatsunori Hashimoto. Reasoning to learn from latent thoughts, 2025. URL https://arxiv.org/abs/2503.18866.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Zafir Stojanovski, Oliver Stanley, Joe Sharratt, Richard Jones, Abdulhakeem Adefioye, Jean Kaddour, and Andreas Köpf. Reasoning Gym: Reasoning environments for reinforcement learning with verifiable rewards. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2025. arXiv:2505.24760.

Fahim Tajwar, Guanning Zeng, Yueer Zhou, Yuda Song, Daman Arora, Yiding Jiang, Jeff Schneider, Ruslan Salakhutdinov, Haiwen Feng, and Andrea Zanette. Maximum likelihood reinforcement learning, 2026. URL https://arxiv.org/abs/2602.02710.

Yunhao Tang, Sid Wang, Lovish Madaan, and Rémi Munos. Beyond verifiable rewards: Scaling reinforcement learning for language models to unverifiable data, 2025. URL https://arxiv.org/abs/2503.19618.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

Jiaxin Wen, Zachary Ankner, Arushi Somani, Peter Hase, Samuel Marks, Jacob Goldman-Wetzler, Linda Petrini, Henry Sleight, Collin Burns, He He, Shi Feng, Ethan Perez, and Jan Leike. Unsupervised elicitation of language models, 2025. URL https://arxiv.org/abs/2506.10139.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, 2022.

Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D. Goodman. Quiet-STaR: Language models can teach themselves to think before speaking. In *Conference on Language Modeling*, 2024.

Jiachen Zhao, Zonghai Yao, Zhichao Yang, and Hong Yu. Large language models are in-context teachers for knowledge reasoning. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 16470–16486, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.961. URL https://aclanthology.org/2024.findings-emnlp.961/.

# A MOTIVATING THE LOG TRANSFORM

This appendix motivates the log transform of Section 2.1. We start from the plain success rate as a reward, view the gen-rl rollout as a conditional latent variable model, and arrive at the training objective ultimately given in the main text ($J_{\text{gen}}$ in (2)) as a smoothed surrogate for the marginal log-likelihood of downstream success.

**Plain success rate as an objective.** In RLVR, we might define the reward over the two-turn gen-rl rollout in terms of the verifier bit, $R = c(A_2) \in \{0, 1\}$, or, for a set of $M > 1$ probes, as the average success rate, $R(\{A_2^m\}) = \frac{1}{M}\sum_{m=1}^{M} c(A_2^m)$, as given in the main text. Using more than one probe leaves the objective unchanged, since the expected reward is the same for every $M$; what it changes is the variance, giving a lower-variance estimate of how well probes are solved with $A_1$ in context.

Using the plain success rate as the reward gives the following objective, written on the two-turn rollout:

$$J_{\text{plain}}(\theta) \;=\; \mathbb{E}_{A_1\sim\pi_\theta(\cdot|Q_1)}\,\mathbb{E}_{\{Q_2^m\}\sim\mathcal{D}'}\,\mathbb{E}_{\{A_2^m\}\sim\pi_\theta(\cdot|Q_1,A_1,Q_2^m)}\big[\,R(\{A_2^m\})\,\big].$$

Its gradient is the usual score-function (REINFORCE) estimator:

$$\begin{aligned}\nabla_\theta J_{\text{plain}} = \mathbb{E}_{A_1\sim\pi_\theta(\cdot|Q_1)}\,\mathbb{E}_{\{Q_2^m\}\sim\mathcal{D}'}\,\mathbb{E}_{\{A_2^m\}\sim\pi_\theta(\cdot|Q_1,A_1,Q_2^m)}\\ \Big[\,R(\{A_2^m\})\Big(\nabla_\theta\log\pi_\theta(A_1\mid Q_1)+\textstyle\sum_m\nabla_\theta\log\pi_\theta(A_2^m\mid Q_1,A_1,Q_2^m)\Big)\Big].\end{aligned} \tag{7}$$

Davis & Recht (2025) show that many approaches to reinforcement learning on language models with binary rewards can be viewed as optimizing some monotonically increasing function of the probability of getting a correct answer to a prompt. Define our success event, $C$, as the event that a single probe, drawn from $\mathcal{D}'$ and answered with the prefix in context, is solved. Then $J_{\text{plain}}$ can be understood as simply maximizing the marginal likelihood of $C$:

$$\begin{aligned}J_{\text{plain}} &= \mathbb{E}_{A_1}\,\mathbb{E}_{Q_2}\,\mathbb{E}_{A_2}\big[\,c(A_2)\,\big]\\ &= \iiint \pi_\theta(A_1\mid Q_1)\,\mathcal{D}'(Q_2)\,\pi_\theta(A_2\mid Q_1,A_1,Q_2)\;c(A_2)\;dA_2\,dQ_2\,dA_1 \;=:\; p_\theta(C\mid Q_1).\end{aligned}$$

That is, $J_{\text{plain}}$ attempts to maximize something like the success of the model at an average question, drawn at random from the data distribution.

**Latent variable model.** An alternative is to view the gen-rl setup through the lens of a latent variable model (Zelikman et al., 2022; Phan et al., 2023; Ruan et al., 2025; Gandhi et al., 2026; Tang et al., 2025). Our data consists in a standard training set of independent (single) examples in isolation, each with a verifiably correct answer. When we arrange these examples in the form of the two-turn rollouts described above, where one answer as a demonstration influences downstream responses, we get a conditional latent variable model that captures the in-context learning effect of $Q_1$ and $A_1$ on $A_2$:

$$p(A_1, A_2 \mid Q_1, Q_2; \theta, \mathcal{D}, \mathcal{D}') \;=\; p(A_1 \mid Q_1; \theta, \mathcal{D})\; p(A_2 \mid Q_1, A_1, Q_2; \theta, \mathcal{D}, \mathcal{D}'),$$

where $\theta$ represents the parameters of our model.

In the latent variable framework, our typical goal would be to fit the parameters $\theta$ by maximizing the marginal log-likelihood of a correct probe answer, $A_2$:

$$J_{\text{MLL}}(\theta) \;=\; \log\mathbb{E}_{A_1\sim\pi_\theta(\cdot|Q_1)}\,\mathbb{E}_{\{Q_2^m\}\sim\mathcal{D}'}\,\mathbb{E}_{\{A_2^m\}\sim\pi_\theta(\cdot|Q_1,A_1,Q_2^m)}\big[\,R(\{A_2^m\})\,\big] \;=\; \log p_\theta(C\mid Q_1).$$

That is, $J_{\text{MLL}}$ is the log of the marginal likelihood given as $J_{\text{plain}}$.

From the standpoint of a single question, $J_{\text{MLL}}$ is just a monotone transform of $J_{\text{plain}}$, and there would be no reason to prefer one over the other (Davis & Recht, 2025). However, the two objectives aggregate over a finite set of data differently. $J_{\text{MLL}}$, when aggregated over a finite dataset, places far more weight on the questions the model rarely gets right – that is, on hard questions.[3] When the

[3]Over $N$ training questions with per-question marginal likelihoods $p_n := p_\theta(C \mid Q_1^n)$, $J_{\text{plain}}$ is the arithmetic mean of the $p_n$. That is, it is the probability that the model succeeds on a question drawn uniformly at random. $J_{\text{MLL}}$, when aggregated over the same finite dataset, gives the log of their geometric mean, $\frac{1}{N}\sum_n \log p_n = \log\big(\prod_n p_n\big)^{1/N}$. The product $\prod_n p_n$ is the probability of succeeding on every question at once, so maximizing $J_{\text{MLL}}$ is maximum-likelihood estimation on the dataset as a whole.

target is problems more difficult than those in the training set, it may be that the reweighting towards hard questions given by the log version of the objective may allocate the training signal more where we would more like it to go.

As in latent-variable models generally, the disadvantage of $J_{\mathrm{MLL}}$ is that it cannot be optimized directly by the sampling-based policy-gradient methods that handle $J_{\mathrm{plain}}$.[4]

What we can do is borrow from variational inference, and "push the log in" using Jensen's inequality. For the unsmoothed reward, this gives a formal lower bound on $J_{\mathrm{MLL}}$ that can be optimized using the same machinery as standard RLVR. Starting from $J_{\mathrm{MLL}}$ in the form above, Jensen's inequality gives, for each $Q_1$,

$$\begin{aligned} J_{\mathrm{MLL}}(\theta) &= \log \mathbb{E}_{A_1\sim\pi_\theta(\cdot|Q_1)}\, \mathbb{E}_{\{Q_2^m\}\sim\mathcal{D}'}\, \mathbb{E}_{\{A_2^m\}\sim\pi_\theta(\cdot|Q_1,A_1,Q_2^m)}\big[\, R(\{A_2^m\})\,\big] \\ &\geq\ \mathbb{E}_{A_1\sim\pi_\theta(\cdot|Q_1)}\, \mathbb{E}_{\{Q_2^m\}\sim\mathcal{D}'}\, \mathbb{E}_{\{A_2^m\}\sim\pi_\theta(\cdot|Q_1,A_1,Q_2^m)}\big[\, \log R(\{A_2^m\})\,\big] \\ &=\ \mathbb{E}_{A_1\sim\pi_\theta(\cdot|Q_1)}\, \mathbb{E}_{\{Q_2^m\}\sim\mathcal{D}'}\, \mathbb{E}_{\{A_2^m\}\sim\pi_\theta(\cdot|Q_1,A_1,Q_2^m)}\Big[\, \log \tfrac{1}{M}\textstyle\sum_m c(A_2^m)\,\Big]. \end{aligned} \tag{8}$$

The last line recalls that the reward is the success rate on the probe set. The problem is that this success rate can be 0 (whenever the model fails to answer any of the $M$ questions correctly) and $\log 0$ is undefined. This results in the smoothing described in Section 2.1 of the main text, which is Laplace smoothing with pseudo-count $\alpha$: $\widetilde{R}(s) = (s+\alpha)/(M+2\alpha)$, the success rate after adding $\alpha$ pseudo-successes and $\alpha$ pseudo-failures, with $\alpha = 1$ throughout. Substituting the smoothed average $\widetilde{R}(s)$ for the naive average in the last line of (8) gives exactly $J_{\mathrm{gen}}$ of (2), the objective we use in practice.

The smoothing step changes the quantity inside the expectation: Jensen's inequality bounds $\log \mathbb{E}[\widetilde{R}]$, not the original $\log \mathbb{E}[R]$. Unless an additional relation between $\mathbb{E}[\widetilde{R}]$ and $\mathbb{E}[R]$ is established, the smoothed objective should therefore be understood as a finite, biased surrogate rather than an exact lower bound on $J_{\mathrm{MLL}}$.

**Choosing between $J_{\mathrm{plain}}$ and $J_{\mathrm{gen}}$.** Compare (7) and (3). The two gradients have the same form. Each is the standard two-turn policy gradient, with a single scalar reward multiplying the log-probabilities of the prefix and of every probe response, and they differ only in that reward: the probe-set success rate for $J_{\mathrm{plain}}$, and the log of its smoothed value for $J_{\mathrm{gen}}$. In each case, both the inner and outer traces (the $A_1$ prefixes and the $A_2$ probe responses) receive the reward, giving rise to the "teacher and student" structure described in the introduction.

Ultimately, we treat the choice between the objectives for present purposes as an empirical one. Table 5 compares the two objectives on multiplication. The two arms share data, probes, model, and training settings, except for the reward and, following Appendix C, the scaling of the format term: fixed $\delta$ under the linear objective, whose advantages sit on a fixed scale, and rescaled per group under the log objective, whose advantages do not. The two arms are indistinguishable on the trained tiers, while on the two held-out tiers the log objective is ahead by 9.4 and 8.3 points (79.3% against 69.9%, and 31.9% against 23.5%), both significant.

## B Derivation of the probe advantage

Recall the part of the gradient we want to estimate for the probe responses, from (5), and the definitions of Section 2.2: $s_i^{-m} := s_i - c_i^m$ is the success count of row $i$ with probe $m$ left out, and $\Delta_i^m := \log \widetilde{R}(s_i^{-m}+1) - \log \widetilde{R}(s_i^{-m})$ is the increase in the row's reward from one more solved probe, evaluated with probe $m$ left out, so that $\log \widetilde{R}(s_i) = \log \widetilde{R}(s_i^{-m}) + c_i^m\, \Delta_i^m$. We can now

[4] Its gradient is $\nabla_\theta\, p_\theta(C \mid Q_1)$ divided by $p_\theta(C \mid Q_1)$, and each of these is an expectation over the rollout; the ratio of two expectations has no unbiased sample estimate.

Table 5: Objective ablation on multiplication (cross-tier): the plain success rate ($J_{\text{plain}}$, "linear") against the log objective ($J_{\text{gen}}$), with the standard control for reference. The log arm is the gen-rl arm of Table 1; the linear arm uses the fixed format weight $\delta$ rather than the per-group rescaling (Appendix C). Gain ratios are each objective's share of the control's gain over the base model, as defined in Section 3.2, and shaded as in Table 1; a star marks a 95% paired bootstrap interval that excludes 0 for $\Delta$ or 1 for the ratio.

| | correctness (%) | | | $\Delta$ (pp) | gain ratio vs control | |
|---|---|---|---|---|---|---|
| tier | control | linear | **log** | log − linear | linear | log |
| $2\times2 \cdot 2\times3$ (ID) | 96.4 | 97.3 | 97.3 | −0.1 | 1.01 | 1.01 |
| $3\times3$ (ID) | 90.3 | 93.9 | 93.9 | +0.0 | 1.04* | 1.04* |
| $3\times4$ (OOD+1) | 72.5 | 69.9 | 79.3 | +9.4* | 0.96 | 1.09* |
| $4\times4 \cdot 4\times5$ (OOD+2) | 19.9 | 23.5 | 31.9 | +8.3* | 1.19* | 1.61* |

rewrite the gradient in terms of $\Delta_i^m$:

$$\begin{aligned}
\nabla_\theta^{\text{probe}} J_{\text{gen}} &= \sum_{m=1}^{M} \mathbb{E}\Big[ \log \widetilde{R}(s_i)\, \nabla_\theta \log \pi_\theta(A_2^{i,m} \mid Q_1, A_1^i, Q_2^m) \Big] \\
&= \sum_{m=1}^{M} \mathbb{E}\Big[ \big(\log \widetilde{R}(s_i^{-m}) + c_i^m\, \Delta_i^m\big)\, \nabla_\theta \log \pi_\theta(A_2^{i,m} \mid Q_1, A_1^i, Q_2^m) \Big] \\
&= \sum_{m=1}^{M} \mathbb{E}\Big[ \log \widetilde{R}(s_i^{-m})\, \nabla_\theta \log \pi_\theta(A_2^{i,m} \mid Q_1, A_1^i, Q_2^m) \Big] \\
&\quad + \sum_{m=1}^{M} \mathbb{E}\Big[ c_i^m\, \Delta_i^m\, \nabla_\theta \log \pi_\theta(A_2^{i,m} \mid Q_1, A_1^i, Q_2^m) \Big]. \qquad (9)
\end{aligned}$$

Take the first addend in (9). Let $\mathcal{C}$ be everything sampled except $A_2^{i,m}$ itself,

$$\mathcal{C} \;:=\; \Big(Q_1,\; A_1^{1:G},\; Q_2^{1:M},\; \{A_2^{j,m'}\}_{(j,m')\neq(i,m)}\Big).$$

Then

$$\begin{aligned}
&\mathbb{E}\Big[ \log \widetilde{R}(s_i^{-m})\, \nabla_\theta \log \pi_\theta(A_2^{i,m} \mid Q_1, A_1^i, Q_2^m) \Big] \\
&\quad = \mathbb{E}_{\mathcal{C}}\Big[ \mathbb{E}_{A_2^{i,m}\mid\mathcal{C}}\Big[ \log \widetilde{R}(s_i^{-m})\, \nabla_\theta \log \pi_\theta(A_2^{i,m} \mid Q_1, A_1^i, Q_2^m) \Big] \Big] \\
&\quad = \mathbb{E}_{\mathcal{C}}\Big[ \log \widetilde{R}(s_i^{-m})\; \underbrace{\mathbb{E}_{A_2^{i,m}\mid\mathcal{C}}\Big[ \nabla_\theta \log \pi_\theta(A_2^{i,m} \mid Q_1, A_1^i, Q_2^m) \Big]}_{=\,0 \text{ (score identity)}} \Big] \\
&\quad = 0.
\end{aligned}$$

We can now drop the first addend from (9), giving

$$\begin{aligned}
\nabla_\theta^{\text{probe}} J_{\text{gen}} &= \sum_{m=1}^{M} \mathbb{E}\Big[ c_i^m\, \Delta_i^m\, \nabla_\theta \log \pi_\theta(A_2^{i,m} \mid Q_1, A_1^i, Q_2^m) \Big] \\
&= \sum_{m=1}^{M} \mathbb{E}\Big[ \big(c_i^m - b_{i,m}\big)\, \Delta_i^m\, \nabla_\theta \log \pi_\theta(A_2^{i,m} \mid Q_1, A_1^i, Q_2^m) \Big]
\end{aligned}$$

for any $b_{i,m}$ that does not depend on the sampled answer $A_2^{i,m}$ itself.

This form of the estimand suggests a baseline parallel to the standard GRPO-style group-relative advantage: center each correctness bit on the model's other attempts at the same probe,

$$\nabla_\theta^{\text{probe}} J_{\text{gen}} \;=\; \sum_{m=1}^{M} \mathbb{E}\Big[ \Big(c_i^m - \frac{1}{G-1}\sum_{j\neq i} c_j^m\Big)\, \Delta_i^m\, \nabla_\theta \log \pi_\theta(A_2^{i,m} \mid Q_1, A_1^i, Q_2^m) \Big].$$

The coefficient on the score function is the probe advantage (6); substituting $\widetilde{R}$ into the definition of $\Delta_i^m$ gives the closed form $\Delta_i^m = \log\big(1 + 1/(s_i^{-m} + \alpha)\big)$.

# C THE FORMAT TERM

Let $F(A) \in \{0, 1\}$ indicate that a response follows the task's answer format (a delimited reasoning-then-answer envelope on the parametric tasks, a boxed final answer on competition math). The format term added to the prefix advantage is $\delta$ times the group leave-one-out contrast of $F(A_1^i)$; the term added to the probe advantage is $\delta$ times $\frac{1}{M}$ times the same-probe column leave-one-out contrast of $F(A_2^{i,m})$, the $\frac{1}{M}$ matching the per-probe share of the row-level reward. We use $\delta = 0.1$.

Under the log objective the advantages are not on a fixed scale: the slope of $\log \widetilde{R}$ is $1/\widetilde{R}$, which ranges over a factor of $(M + \alpha)/\alpha$ between all-wrong and all-right rows, or nine at $M = 8$ and $\alpha = 1$. A fixed $\delta$ is therefore a much larger fraction of the advantage on saturated groups than on hard ones. On the parametric tasks we rescale the format contrast per group and per term so that its spread is a fixed fraction $\delta$ of the spread of the advantage it accompanies, $\mathcal{A}_i^1$ or $\mathcal{A}_{i,m}^2$. On competition math the fixed $\delta$ was used as is.

# D EXPERIMENTAL DETAILS

This appendix gives the settings behind the runs in Tables 1–4.

## D.1 SETTINGS BY TASK TYPE

Table 6 lists the settings that are properties of each particular testbed.

## D.2 SETTINGS BY ARM TYPE

Table 7 lists the settings that are properties of the arm type rather than of the task.

Estimator settings for the standard RLVR controls were driven primarily by library defaults, with some concessions to DAPO-style optimizations (Yu et al., 2025). It was not possible to port all estimator settings from the control to gen-rl. Most importantly, std. normalization is inconsistent with the log rescaling of our $\log \widetilde{R}(s)$ reward. Rescaling each group to unit spread would remove the between-group component of the weighting that the log reward introduces (Section 2.1). To evaluate whether estimator differences account for all or part of the effect reported in the main text, we ran an ablation in which we retrained the knights & knaves control with the three estimator settings conformed to gen-rl's choices (leave-one-out baseline, no standard-deviation normalization, token sum$/L$ with $L$=2048). The estimator-matched control departs significantly from the standard control at two OOD tiers, and in both such tiers the standard control is ahead (Table 8). Gen-rl's margin over the matched control is significant on every tier (Table 8, right). We report the stronger standard control settings in the main paper.

## D.3 PROMPTS AND ROLLOUT FORMAT

Figure 3 shows examples of the question text for the parametric tasks. On those tasks the question is the user message of a single-turn chat that opens with the system prompts below:

```
You are solving multiplication problems.  Think step by step, then
give your final answer.

Respond in the following format:
<reasoning>
...
</reasoning>
<answer>
...
</answer>
```

Table 6: **Settings by task type.** Each value applies to both the control and the gen-rl arm of the task. For competition math, a value in parentheses is the compute-matched control's where it differs from the other two arms. Evaluation items and attempts per cell are in Table 9.

| | Multiplication | Knights & knaves | Competition math |
|---|---|---|---|
| *Model* | | | |
| base model | Qwen2.5-1.5B-Instruct | Qwen2.5-1.5B-Instruct | Qwen3-1.7B |
| fine-tuning | LoRA, rank 8, $\alpha$=32 | LoRA, rank 8, $\alpha$=32 | full |
| Qwen3 thinking mode | — | — | off |
| *Training data* | | | |
| training tiers | 2×2, 2×3, 3×3 | 2, 3, 4 inhabitants | bands 2–7 |
| training prompts (one epoch) | 6,000 | 6,000 | 12,000 |
| prompts per step | 12 | 12 | 24 |
| steps | 500 | 500 | 500 |
| *Rollouts* | | | |
| rollouts per prompt, $G$ | 12 | 12 | 8 (72) |
| probes per prefix, $M$ (gen-rl) | 8 | 8 | 8 |
| temperature | 1.0 | 1.0 | 1.0 |
| top-$p$ | 1.0 | 1.0 | 1.0 |
| max prompt tokens | 1,024 | 1,024 | 2,560 |
| max generated tokens | 1,024 | 1,024 | 4,096 |
| *Optimizer (AdamW)* | | | |
| learning rate | $10^{-5}$ | $10^{-5}$ | $10^{-6}$ |
| schedule | constant | constant | constant |
| weight decay | 0 | 0 | 0 |
| $(\beta_1, \beta_2)$ | (0.9, 0.999) | (0.9, 0.999) | (0.9, 0.999) |
| gradient clip | 1.0 | 1.0 | 1.0 |
| seed | 42 | 42 | 42 |
| *Compute* | | | |
| GPUs per run | 4 | 4 | 4 (8) |
| *Evaluation* | | | |
| temperature | 1.0 | 1.0 | 0.6 |
| top-$p$ | 1.0 | 1.0 | 1.0 |
| max generated tokens | 1,024 | 1,024 | 4,096 |

Table 7: **Settings by arm type.** "Token sum / $L$" follows Dr. GRPO (Liu et al., 2025). "Over-long filtering" masks a response that hits the generation cap out of the loss. "Format-term scaling" is defined in Appendix C.

| | control | gen-rl, parametric tasks | gen-rl, competition math |
|---|---|---|---|
| reward | correctness | $\log \widetilde{R}(s)$, $\alpha$=1 | $\log \widetilde{R}(s)$, $\alpha$=1 |
| format weight $\delta$ | 0.1 | 0.1 | 0.1 |
| format-term scaling | fixed | per group | fixed |
| advantage baseline | group mean | leave-one-out | leave-one-out |
| std. normalization | yes | no | no |
| loss aggregation | token mean | token sum / $L$ | token sum / $L$ |
| $L$, loss normalizer (tokens) | — | 2,048 | 16,384 |
| over-long filtering | on | on | off |
| PPO clip $\epsilon$ | 0.2 | 0.2 | 0.2 |
| KL / entropy terms | none | none | none |
| updates per step | 1 | 1 | 1 |

Table 8: **A knights & knaves control trained with gen-rl's estimator settings.** "Matched control" is the standard control of Table 2 retrained with the leave-one-out baseline, no standard-deviation normalization and token sum$/L$; everything else is unchanged. Correctness at step 500 on the items of Table 2; $\Delta$ (pp) and gain ratios with the conventions of Table 1. The gain ratio is (gen-rl $-$ base) / (arm $-$ base) for the arm named in the column head, so the right-hand column is gen-rl's gain in units of the matched control's.

| | correctness (%) | | | | $\Delta$ (pp) | | gain ratio, gen-rl | |
|---|---|---|---|---|---|---|---|---|
| tier | base | control | matched control | **gen-rl** | matched $-$ control | gen-rl $-$ matched | vs control | vs matched |
| 2 inhabitants (ID) | 0.6 | 53.5 | 50.9 | 63.5 | $-2.7$ | $+12.6^*$ | $1.19^*$ | $1.25^*$ |
| 3 inhabitants (ID) | 0.3 | 33.9 | 35.1 | 50.7 | $+1.2$ | $+15.6^*$ | $1.50^*$ | $1.45^*$ |
| 4 inhabitants (ID) | 0.0 | 23.7 | 22.1 | 39.8 | $-1.5$ | $+17.7^*$ | $1.68^*$ | $1.80^*$ |
| 5 inhabitants (OOD+1) | 0.0 | 16.4 | 14.5 | 27.3 | $-1.9$ | $+12.8^*$ | $1.67^*$ | $1.88^*$ |
| 6 inhabitants (OOD+2) | 0.1 | 11.2 | 7.6 | 21.4 | $-3.6^*$ | $+13.8^*$ | $1.92^*$ | $2.83^*$ |
| 7 inhabitants (OOD+3) | 0.0 | 6.5 | 4.9 | 14.5 | $-1.6^*$ | $+9.5^*$ | $2.21^*$ | $2.93^*$ |
| 8 inhabitants (OOD+4) | 0.0 | 2.1 | 2.1 | 12.6 | $+0.1$ | $+10.5^*$ | $6.10^*$ | $5.91^*$ |

```
You are solving logic puzzles about truth-tellers and liars.
Think step by step, then give your final answer.

Respond in the following format:
<reasoning>
...
</reasoning>
<answer>
...
</answer>
```

On competition math there is no system prompt: the user message is the problem statement followed by "`Let's think step by step and output the final answer within \boxed{}.`", and the answer is the last boxed expression. Each model is prompted through its own chat template; for Qwen3 the template is used with thinking disabled.

Probe responses are sampled with the same temperature and generation cap as prefixes (Table 6). Evaluation uses the single-turn chat only.

### D.4 Training data and probes

**Parametric tasks.** A multiplication question of tier $d_1 \times d_2$ draws each operand uniformly from the integers with that many digits. The 6,000 training questions come in equal proportions from the two training tiers, $\{2\times2, 2\times3\}$ and $3\times3$, with the digit pair drawn uniformly within the first; operand pairs are assigned to training or evaluation by a fixed hash, so no evaluation question is a training question. Knights & knaves puzzles are generated by Reasoning Gym (Stojanovski et al., 2025) with the number of inhabitants fixed per tier, and the 6,000 training puzzles come in equal proportions from the three training tiers. The model sees the full generated puzzle, including its island preamble and its answer-format instruction, which Figure 3 omits; the generator also varies the names of the two roles from puzzle to puzzle (knights and knaves, saints and sinners, and so on), and the reference answer uses the puzzle's own names. Evaluation puzzles are generated from a seed range disjoint from that of the training puzzles.

**Competition math.** Section 3.1 describes the bands. Before training, 5% of each trained band was set aside as the ID evaluation set, and the 12,000 training questions were drawn once, without replacement, from the remaining problems in bands 2–7, so their mix of bands is that of the pool. The near-OOD, far-OOD and retention sets are frozen draws from bands 1, 0 and 8 (Table 9).

**Probes.** Section 3.1 gives the rule for the probe distribution $\mathcal{D}'$. In detail, the tiers are the training tiers of Table 6, and for a $Q_1$ from any tier but the hardest, each of the $M$ probes is drawn from a uniformly chosen harder training tier, independently and with replacement; for a $Q_1$ from the hardest

training tier, the probes are drawn from that tier itself, excluding $Q_1$. So every multiplication $Q_1$ is probed with $3\times3$ questions; a 2-inhabitant puzzle with a mix of 3- and 4-inhabitant puzzles, and 3- and 4-inhabitant puzzles with 4-inhabitant ones; and a competition-math question with questions from harder trained bands, band 2 with band 2. Probes are drawn from the training pool only, never from an evaluation set. They are drawn once, when the training set is built, so all $G$ rollouts of a $Q_1$ share the same $M$ probes (Section 2.2).

### D.5 Evaluation protocol

Table 9 lists the number of held-out items and attempts per item behind every cell of Tables 1–4: parametric-task cells use 500 fresh items per tier at three attempts each, and competition-math cells use frozen draws from the held-out pools. Every arm, and the base model, is evaluated on the same items, prompted in the same format as for a first-turn question during training, with the sampling settings in the evaluation rows of Table 6; the intervals in the tables are paired bootstraps over items, as described in the caption of Table 1.

Correctness is the training verifier applied to a single response. On multiplication the content of the answer tag must parse to a number equal to the product; on knights & knaves it must state the full assignment exactly, with no partial credit; on competition math the last boxed expression must be equivalent to the reference answer. A response that reaches the generation cap, or that has no extractable answer, counts as incorrect, and the format term plays no part in evaluation.

Table 9: Items $\times$ attempts per evaluation cell.

| task | tier / leg | items $\times$ attempts |
|---|---|---|
| multiplication | $2\times2 \cdot 2\times3$ (ID) | $500\times3$ |
| | $3\times3$ (ID) | $500\times3$ |
| | $3\times4$ (OOD+1) | $500\times3$ |
| | $4\times4 \cdot 4\times5$ (OOD+2) | $500\times3$ |
| knights & knaves | 2 inhabitants (ID) | $500\times3$ |
| | 3 inhabitants (ID) | $500\times3$ |
| | 4 inhabitants (ID) | $500\times3$ |
| | 5 inhabitants (OOD+1) | $500\times3$ |
| | 6 inhabitants (OOD+2) | $500\times3$ |
| | 7 inhabitants (OOD+3) | $500\times3$ |
| | 8 inhabitants (OOD+4) | $500\times3$ |
| competition math | ID holdout (bands 2–7) | $1089\times4$ |
| | near-OOD (band 1) | $400\times8$ |
| | far-OOD (band 0) | $500\times8$ |
| | retention (band 8) | $150\times4$ |
| external benchmarks | AIME 2024 | $30\times16$ |
| | AIME 2025 | $30\times16$ |
| | POLARIS near-OOD (band 1) | $400\times8$ |
| | POLARIS far-OOD (band 0) | $500\times8$ |
| | POLARIS retention (band 8) | $150\times4$ |

### D.6 Compute matching and software

Section 3.1 describes the compute-matched control for competition math. We matched the number of trained sequences per prompt rather than the batch size: matching by batch size at a fixed number of steps would have required nine epochs over the 12,000-question stream, which we judged an undue risk of overfitting. The parametric tasks have a standard control only.

All runs use SkyRL v0.2.0 (Cao et al., 2025), with vLLM 0.19.0 (Kwon et al., 2023) for rollouts and PyTorch 2.10 FSDP2 for training; LoRA (Hu et al., 2022) fine-tuning applies rank-8 adapters to every linear layer of the transformer blocks. Knights & knaves puzzles come from Reasoning Gym (Stojanovski et al., 2025), and competition-math answers are graded with Math-Verify.[5] The

[5] https://github.com/huggingface/Math-Verify

gen-rl rollout, the probe draw and the advantages of Section 2.2 are implemented as an extension of SkyRL's generator; the control arms use SkyRL's GRPO implementation with the settings in Table 7.

## E TRAINING CURVES FOR EVERY TIER

Figure 4 shows how gen-rl's advantage over standard RLVR unfolds over training, on the hardest evaluation tier of each task that has intermediate checkpoints. Figure 5 repeats it for every evaluation tier with intermediate checkpoints: every multiplication tier, knights & knaves with 3 to 6 inhabitants, and every competition-math leg. The parametric tasks were evaluated every 50 steps; the competition-math far-OOD leg every 25 steps, and its other three legs at steps 250 and 500 only.

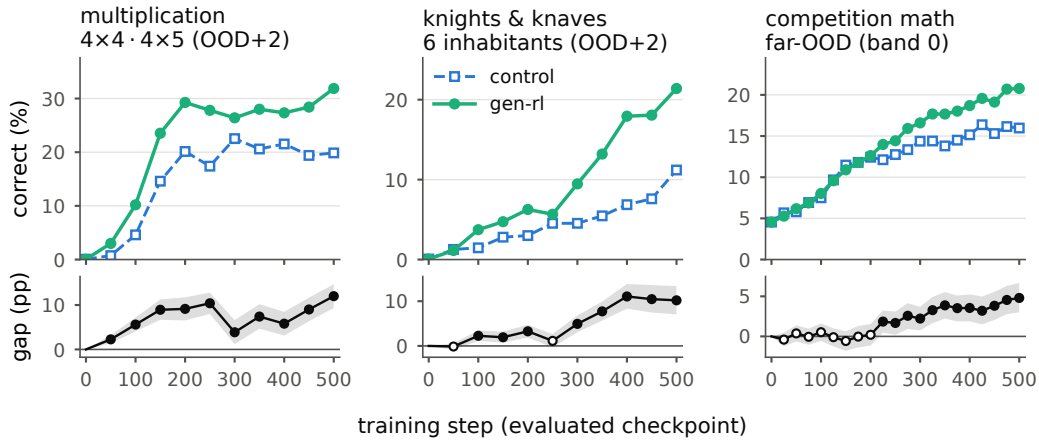



Figure 4: **Evaluation curves.** These curves show the results on held-out evaluation problems as training progresses. We save checkpoints at 50-step intervals for the parametric tasks, and at 25-step intervals for competition math. We evaluate each checkpoint using the same evaluation sets as reported in the main results tables as of the last step of training. The strip under each panel is the paired per-item gap, gen-rl − control, in percentage points, with its 95% bootstrap interval; a filled marker means the interval excludes 0. For multiplication and competition math, the figure shows the hardest tier of evaluation problem available. The knights & knaves panel shows 6 inhabitants, the farthest tier with intermediate checkpoints (7 and 8 were evaluated at step 500 only); every tier is in Figure 5.

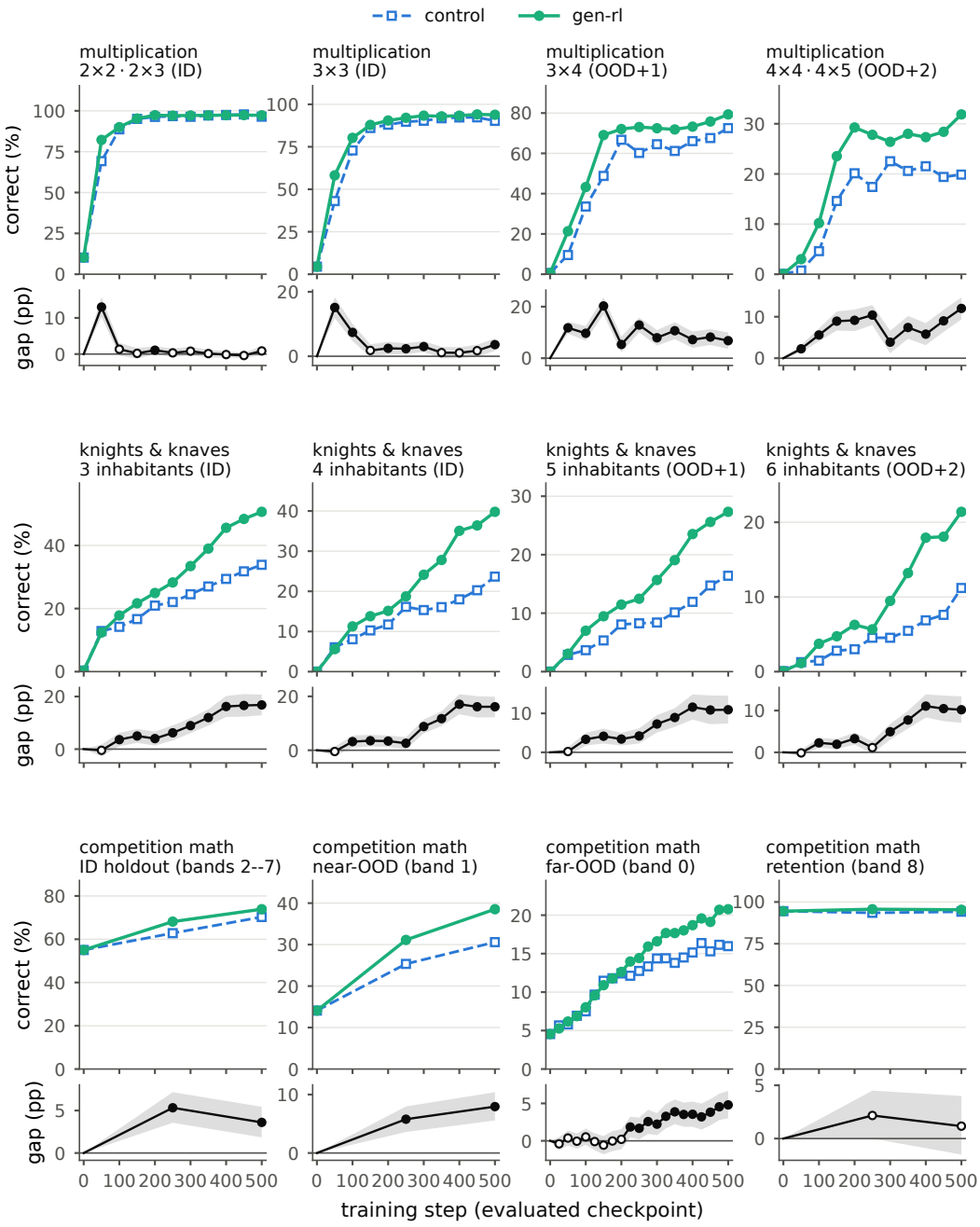



Figure 5: Held-out correctness over training for every tier and leg, with the paired gap strip of Figure 4 under each panel. Conventions as in Figure 4.