WHAT ACTUALLY EVOLVES IN SELF-EVOLVING REINFORCEMENT LEARNING?

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-evolving reinforcement learning (RL) aims to train a model that learns from self-generated tasks, without additional human annotations. It typically alternately trains two roles: a proposer that generates training tasks and a solver that learns from them, spontaneously improving the performance on downstream tasks. However, it remains unclear what drives these gains, and whether the proposer can keep generating tasks that sustain the solver's improvement. In this study, we systematically examine current self-evolving RL across *six* methods and *six* models on *seven* reasoning benchmarks, with RL on human-annotated answers (human-annotated RL) as a reference. We track how pass@1 and pass@128 change, identify the source of the gain, and test whether later proposers write more useful tasks than earlier ones. Our analysis reveals three findings. (1) Most solver gains come from questions the base model already answers correctly. (2) Self-evolving RL corrects far fewer errors than human-annotated RL. (3) Proposers do not teach more after further training: the same solver does no better, and sometimes worse, when trained on tasks from a later proposer than on tasks from an earlier one. Our theoretical analysis of a single solver update shows that it can only strengthen answers the solver already gives. Although recursive self-improvement aims to go beyond human supervision, current self-evolving RL rarely outperforms human-annotated RL. Taken together, our findings suggest that current gains primarily reflect sharpening, and that self-evolving RL has so far only partially achieved sustained recursive self-improvement.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) has substantially improved the reasoning capabilities of large language models (LLMs) in tasks such as mathematical reasoning and coding (Shao et al., 2024; DeepSeek-AI, 2025). Despite its effectiveness, RLVR needs high-quality problems and reliable supervision, and preparing such data is costly, which creates a bottleneck for scaling the training (Huang et al., 2026; Zhao et al., 2025a). Self-evolving RL offers a label-free alternative through iterative self-training (Huang et al., 2026; Zhao et al., 2025a; Liu et al., 2025; 2026): a *proposer* generates training tasks for a *solver* to learn from, and both are updated in turn. Such methods now span mathematics, code, tool use, search, and theorem proving (Xia et al., 2026; Yue et al., 2026; Fan et al., 2026; Dong & Ma, 2025) and report strong gains without human-labeled data (Huang et al., 2026; Liu et al., 2025; Xia et al., 2026; Liu et al., 2026; Zhao et al., 2025a). Ideally, as the proposer and solver iteratively optimize each other, an endless cycle is expected to happen that could even lead toward recursive self-improvement (Good, 1965; Schmidhuber, 2007; Zelikman et al., 2024; Zhang et al., 2026).

Yet, the reported improvements deserve careful study. Existing work mainly reports pass@1, which can arise because the solver newly solves questions it could not solve before, or because it answers questions it could already solve more often. This raises two fundamental questions (Figure 1): **(1) What does the solver actually learn? (2) Does the proposer supply more useful training tasks as it evolves?** To answer the first question, we examine how pass@1 and pass@128 change throughout training. We probe whether the pass@1 gain arises on questions the base model gets wrong, suggesting the model learns previously unsolved questions. To answer the second question, we test whether tasks from later proposers teach more than tasks from early ones. Because the tasks and the solver evolve together in the loop, we train copies of the same initial solver on tasks from early and later proposers, ensuring that the difference only derives from the proposers.

Using these tools, we study six self-evolving RL methods, R-Zero (Huang et al., 2026), SPICE (Liu et al., 2025), SPIRAL (Liu et al., 2026), AZR (Zhao et al., 2025a), Agent0 (Xia et al., 2026), and

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.

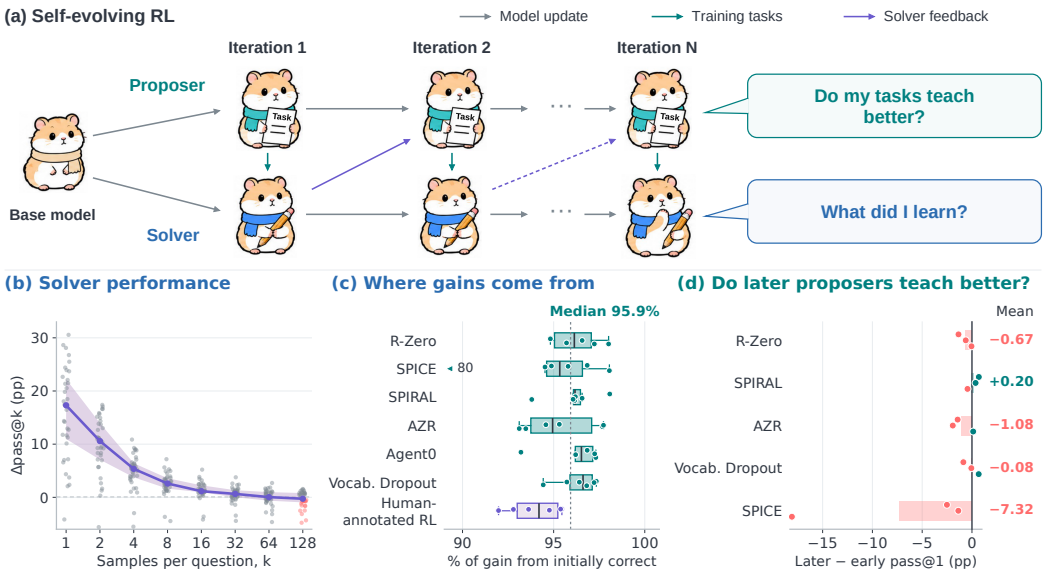

Figure 1: (a) Self-evolving loop. (b) Pass@k change at the peak (LLM judge). (c) Percentage of the pass@1 gain from initially correct questions. (d) Later minus early proposer. pp: percentage points.

Vocabulary Dropout (Dineen et al., 2026), on six different models. We evaluate more than 350 checkpoints with 128 samples on 2,866 questions from seven math benchmarks. As a reference, we also train the same models with GRPO (Shao et al., 2024) on the human-annotated DAPO-Math-17k (Yu et al., 2025) (*human-annotated RL*). We compare each model at initialization with its *peak*, the checkpoint with the highest evaluation pass@1, and call the difference its *gain*. Our findings reveal a gap between current self-evolving RL and sustained self-improvement:

- **Most solver gains come from questions the base model already answers correctly.** Questions whose plurality answer is correct before training make up a median of 73% of the questions but account for a median of 95.9% of the pass@1 gain (Figure 1c), also when an LLM judge scores the answers. The solver mostly answers these questions correctly more often, and improves little on the questions it gets wrong.
- **Self-evolving RL corrects far fewer errors than human-annotated RL.** On questions the base model gets wrong, human-annotated RL raises the percentage of correct responses more than twice as much as self-evolving RL (Figure 4c). Labels from code execution or games (AZR, SPIRAL) keep majority-vote accuracy from falling but do not close this gap.
- **Proposers do not teach more after further training.** In all five methods we test, tasks from later proposers teach a fixed solver no more than tasks from early ones (Figure 1d). Over R-Zero's rounds, more of its questions are invalid and fewer of its labels are correct.

We also present a theoretical analysis of one solver update (Section 5). We show that in a simplified setting, because an update scales the labeled answer's probability, even correct labels barely raise answers the solver rarely gives, majority-vote labels strengthen its top answer, right or wrong, and a proposer changes only which questions are trained on, consistent with our findings.

Together, these results show that the gains of current self-evolving RL mostly reinforce what the base model can already do, and that updating the proposer does not yet supply the new learning signal that sustained self-improvement would require. Recursive self-improvement aims to go beyond human supervision, yet current self-evolving RL rarely outperforms RL on human-annotated answers. This highlights a gap between current methods and the goal of self-evolution, underscoring the need for training signals that correct the solver's errors and for proposer objectives tied to what the solver learns. We will release all code, data, and traces upon acceptance.

2 Preliminaries and Experimental Setup

2.1 Self-Evolving RL

Self-evolving RL trains a model without human-annotated data by letting it generate its own training tasks (Figure 1a). Two roles are trained in alternation. The *proposer* μ_ϕ, called the *challenger* in R-Zero and SPICE (Huang et al., 2026; Liu et al., 2025), is trained to write tasks that are hard but

Table 1: Training experience and reward sources of the study methods.

Method	Training experience	Reward source
R-Zero (Huang et al., 2026)	Generated math questions	Solver majority vote
SPICE (Liu et al., 2025)	Questions from source documents	Document-grounded answers
Agent0 (Xia et al., 2026)	Generated tool-use tasks	Tool-assisted majority vote
Vocabulary Dropout (Dineen et al., 2026)	Questions with vocabulary dropout	Proposer-stated answers
AZR (Zhao et al., 2025a)	Executable reasoning tasks	Execution checks
SPIRAL (Liu et al., 2026)	Self-play games	Game outcomes
Human-annotated RL	Curated math questions	Human-annotated answers

solvable for the current solver. The *solver* π_θ attempts these tasks and is trained on them, and its behavior on them in turn becomes the proposer's reward. One round therefore has four steps: generate tasks, label and filter them, update the solver, and update the proposer (Appendix Algorithm 1). Table 1 lists how each method implements these steps, which we describe below.

Solver update. Given a question q with label z, the solver samples G responses $y_1, \dots, y_G \sim \pi_{\theta_{\text{old}}}(\cdot \mid q)$ with rewards $r_i = R_{\text{solver}}(y_i, z) \in \{0, 1\}$ for whether the final answer matches z. The studied methods train the solver with GRPO (Shao et al., 2024) or a variant of it:

$$\mathcal{J}_{\text{S}}(\theta) = \mathbb{E}\big[\tfrac{1}{G}\sum_{i=1}^{G} \min\big(\rho_i \hat{A}_i,\ \text{clip}(\rho_i, 1-\epsilon, 1+\epsilon)\hat{A}_i\big)\big] - \beta\,\text{KL}\big(\pi_\theta \,\|\, \pi_{\text{ref}}\big), \tag{1}$$

where $\hat{A}_i = (r_i - \text{mean}(\mathbf{r}))/\,\text{std}(\mathbf{r})$ and $\rho_i = \pi_\theta(y_i \mid q)/\pi_{\theta_{\text{old}}}(y_i \mid q)$, applied per token. Responses above the group average become more likely, and the KL term keeps the solver close to a reference model. Methods differ in the source of q and z.

Label construction. In RL on human-annotated data, z is the reference answer. Self-evolving methods must construct z themselves, and this label decides which answers the update rewards. One construction samples m solver answers $a_1, \dots, a_m$ to each generated question and uses their majority vote, the most frequent answer, as the label,

$$z(q) = \arg\max_a \sum_{j=1}^{m} \mathbf{1}\{a_j = a\}, \qquad \kappa(q) = \tfrac{1}{m}\max_a \sum_{j=1}^{m} \mathbf{1}\{a_j = a\}, \tag{2}$$

and keeps q for training only if its agreement $\kappa(q)$, the percentage of votes for that answer, lies in a fixed band (30–80% in R-Zero). The band removes questions that are too easy, where nearly all votes agree, and questions where no answer is clearly preferred. We refer to this construction as *self-voting*; R-Zero and Agent0 use it. A self-voted label is correct only when the solver's most common answer is correct, so on questions the solver usually gets wrong, it rewards a wrong answer. The other methods take the label from outside the solver's vote (Table 1). Either the proposer states an answer when it writes the question, sometimes by extracting it from a source document, and the label is then only as reliable as the proposer; or an environment runs a program or plays out a game, which gives a correct signal, but about code or games rather than mathematics.

Proposer update. The proposer is trained with the same policy-gradient objective, but its reward scores a generated question by the current solver's behavior on it. Most methods reward questions of intermediate difficulty. With self-voting, a common choice rewards questions on which the solver's votes are most split,

$$R_{\text{P}}(q) = 1 - 2\,\big|\kappa(q) - \tfrac{1}{2}\big|, \tag{3}$$

minus a penalty for repeated questions, as in R-Zero. The reward peaks when the votes split evenly ($\kappa = 1/2$) and is zero when they all agree. When both the labels and this reward come from the solver's own answers, the loop needs no human-annotated data.

Related work. Self-evolving RL extends to search, code, and theorem proving (Yue et al., 2026; Fan et al., 2026; Dong & Ma, 2025). Label-free RL trains on majority-vote or confidence rewards (Zuo et al., 2025; Shafayat et al., 2025; Zhao et al., 2025c; Prabhudesai et al., 2025), which can amplify a wrong consensus (He et al., 2026; Lin et al., 2026) or collapse (Lin, 2026), and even RLVR with human labels rarely expands what the base model can solve (Yue et al., 2025; Zhao et al., 2025b; Matsutani et al., 2025). We test these limits when the model writes its own tasks (Appendix A).

2.2 Experimental Setup

Methods and models. We study six self-evolving RL methods: R-Zero (Huang et al., 2026), SPICE (Liu et al., 2025), SPIRAL (Liu et al., 2026), AZR (Zhao et al., 2025a), Agent0 (Xia et al., 2026), and Vocabulary Dropout (Dineen et al., 2026). Each method follows its published training schedule, except SPICE, which has no official code release: we implement it from the

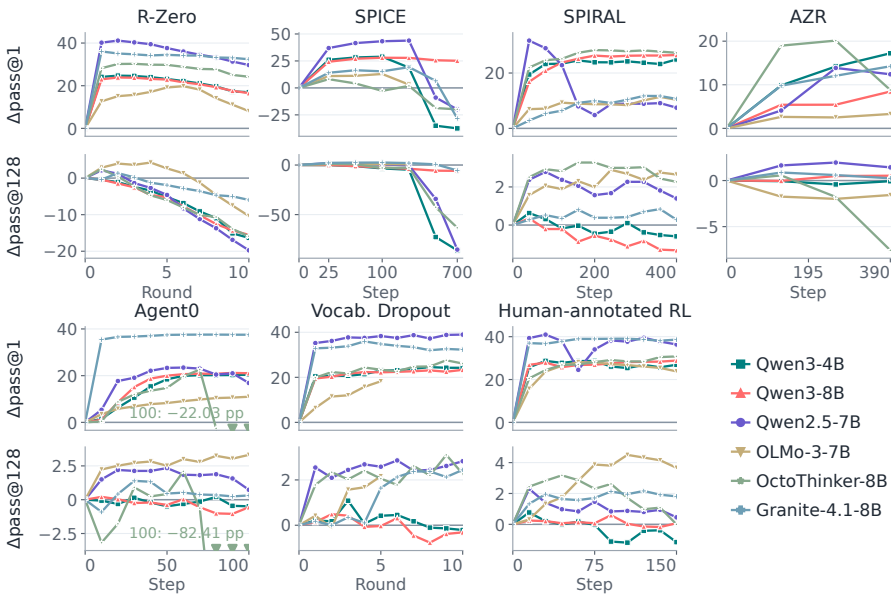

Figure 2: Change in pass@1 and pass@128 from initialization (pp). The SPICE step axis is linear up to step 25 and logarithmic above.

paper, train it for 700 steps (200 for OLMo-3-7B), and select its peak among checkpoints every 100 steps (Appendix E.2). As a reference, *human-annotated RL* trains the same models with GRPO on DAPO-Math-17k (Yu et al., 2025) for 150 steps. We use six base models from four families: Qwen3-4B-Base, Qwen3-8B-Base, Qwen2.5-7B, OLMo-3-7B, OctoThinker-8B-Hybrid-Base, and Granite-4.1-8B-Base. We refer to each method–model pair as a *setting*: 36 self-evolving settings and six for human-annotated RL.

Evaluation protocol. Following R-Zero (Huang et al., 2026), we evaluate on seven mathematical reasoning benchmarks, OlympiadBench-EN (He et al., 2024), MATH-500 (Hendrycks et al., 2021; Lightman et al., 2024), Minerva-Math (Lewkowycz et al., 2022), AMC (MAA, 2023), AIME 2024 and 2025 (MAA, 2024; 2025), and the GSM8K test set (Cobbe et al., 2021): 2,866 questions in total (Appendix E.1.2). For each question we sample $n = 128$ responses with temperature 1.0, top-p 1.0, top-k 40, and a 4,096-token output limit. If c_q of the 128 responses to question q are correct, we report pass@1 $= \frac{1}{N}\sum_q c_q/128$ over the N questions, the average accuracy of a single response, and pass@128 $= \frac{1}{N}\sum_q \mathbf{1}\{c_q > 0\}$, the percentage of questions with at least one correct response (Chen et al., 2021). We refer to the most common final answer among a question's 128 responses, counting equivalent answers as one, as its *plurality answer*, and to the percentage of questions whose plurality answer is correct as *maj@128*. We refer to each setting's best checkpoint across training, as measured by evaluation pass@1, as its *peak*, and to the change from initialization to the peak, in percentage points (pp), as its *gain*.

Grading. The rule-based grader MathRuler (hiyouga, 2025) compares each response's boxed final answer with the reference, and its scores select each setting's peak. Because it misses correct answers that are not boxed, an LLM judge also rescores all responses at initialization and at the peak, accepting a correct final answer in any format but not an unfinished solution (Appendix B.1). All results use MathRuler unless we state that they use the LLM judge.

3 Solver Gains Mainly Reinforce Existing Correct Answers

We now address the first question: what does the solver actually learn? We measure how pass@1 and pass@128 change, identify the source of the gain, and test whether the label source matters.

3.1 How Much the Solver Improves

Figure 2 shows how pass@1 (top rows) and pass@128 (bottom rows) change from initialization during training. Pass@1 rises in all 42 settings, in line with the gains the original papers report (Huang et al., 2026; Zhao et al., 2025a; Liu et al., 2026; Xia et al., 2026; Dineen et al., 2026), but its peak stays below the base model's majority-vote accuracy (maj@128) in every setting. Pass@128 changes little or falls: at the peak, under the LLM judge, it falls in 19 of the 36 self-evolving settings (Figure 1b).

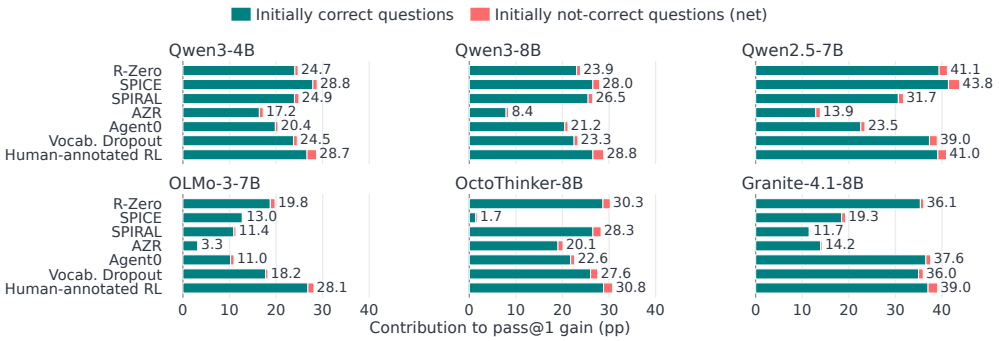

Figure 3: Pass@1 gain by question group (pp)

A higher pass@1 can thus coexist with worse capability on questions the base model could already solve. This raises the question: where does the gain come from?

3.2 Decomposing the Gain by Question Group

Two question groups. Training can raise pass@1 in two ways: by correcting the base model's errors, or by making it more consistent where it is already right. To distinguish between the two, we split the questions by the base model's plurality answer, before any training, so that a question's group does not depend on what training does to it. We refer to questions whose plurality answer is correct as *initially correct questions* and to the rest as *initially not-correct questions* (the plurality answer is wrong or, rarely, missing). Correcting errors shows up as gain on *initially not-correct questions*; becoming more consistent shows up as gain on *initially correct questions*.

The contribution of a group is the total change in the percentage of correct responses over its questions, divided by the number of questions N. With $\hat{p}_t(q)$ the percentage of correct responses among the 128 sampled for question q at checkpoint t, and $S(q)$ its group, the two contributions sum to the pass@1 gain:

$$G_s(t) = \frac{1}{N} \sum_{q:S(q)=s} [\hat{p}_t(q) - \hat{p}_0(q)], \qquad \widehat{\Delta \text{pass@1}}_t = \sum_s G_s(t). \tag{4}$$

Most of the gain comes from questions the base model already answers correctly. Initially correct questions make up a median of 73% of the questions but give a median of **95.9%** of the pass@1 gain (Figure 3). Averaged over the questions in each group, the percentage of correct responses in the self-evolving settings rises by a median of 32.0 pp on initially correct questions but by only 3.2 on initially not-correct questions. Training thus makes the solver more consistent where it is already right, while correcting few of its errors.

Answer formatting accounts for over a quarter of the gain. MathRuler extracts only boxed final answers, so a correct answer in another format counts as wrong, and a solver that learns to box answers it already gives gains accuracy without answering differently. The LLM judge accepts a correct final answer in any format, so rescoring the same responses separates the two: correct answers that MathRuler fails to extract account for a median of 31% of the pass@1 gain in the self-evolving settings, and about as much for human-annotated RL (Figure 4a).

Little of the gain reflects new capability. Formatting does not explain why the gain concentrates on initially correct questions. When the LLM judge, which accepts a correct answer in any format, scores the responses, initially correct questions still give a median of 91% of the gain in the self-evolving settings (Figure 9). Self-evolving RL thus mainly makes the solver give answers it can already find more often, rather than letting it solve new questions, in line with what Yue et al. (2025) report for RL with verifiable rewards.

Checking the reasoning leads to the same conclusion. On AIME, many correct answers come from flawed reasoning, so for Qwen3-8B we also count a question as solved only if at least one correct response reasons validly (CoT-Pass@128; Wen et al., 2026). By this count, the base model solves 16 of the 60 questions, the self-evolving methods at most 22, and human-annotated RL 25 (Figure 5; Appendix B.3).

Human-annotated RL corrects more errors. Human-annotated RL gains mostly on initially correct questions, but on initially not-correct questions it raises the percentage of correct responses by a median of 7.0 pp, more than twice the 3.2 of the self-evolving settings (Figure 4c). The base models can fix more errors; self-evolving RL seems limited by its training signal rather than by the model.

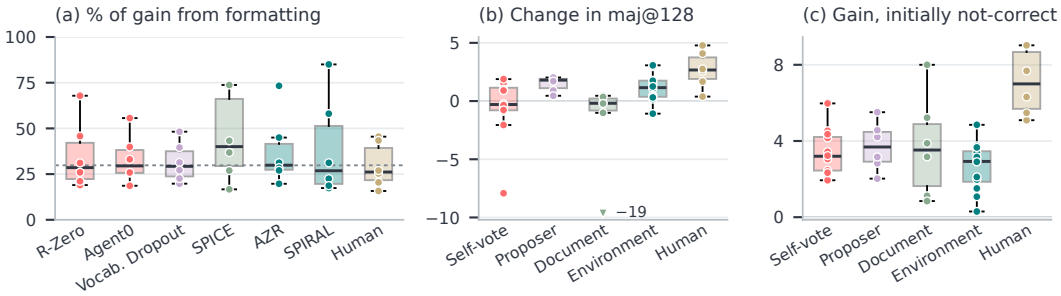

Figure 4: (a) Percentage of the pass@1 gain from answer formatting, by method. (b) Change in maj@128 and (c) gain on initially not-correct questions, by label source (pp; one dot per setting).

3.3 Effect of the Label Source

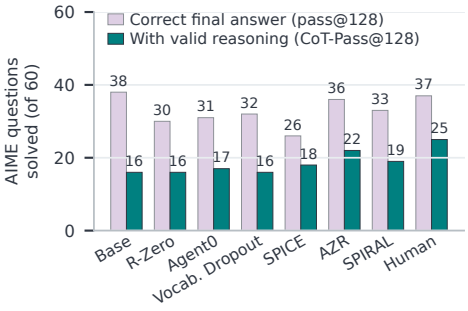

Figure 5: Qwen3-8B on AIME: solved by final answer and with valid reasoning.

Since the label decides which answer an update strengthens (Section 5), we ask whether labels from outside the model change what the solver learns. We compare five label sources (Table 1): self-voting (R-Zero, Agent0), proposer-stated answers (Vocabulary Dropout), answers from documents (SPICE), an environment that runs code or games (AZR, SPIRAL), and human annotation. Pass@1 rises in every setting, so it cannot tell these sources apart; what differs is which answer an update strengthens. We therefore track two measures (Figure 4b,c). The first is maj@128, the percentage of questions whose most common answer is correct. A label usually reinforces that answer, so maj@128 falls when labels strengthen wrong answers and holds or rises when they strengthen correct ones. The second is the gain on initially not-correct questions, which shows whether the solver learns correct answers it did not favor before training, that is, whether it corrects errors.

Outside labels keep correct answers from eroding. With self-voting, a training question is labeled with the solver's own plurality answer, so the solver is also trained on its wrong answers (Section 5); in R-Zero, the percentage of correct labels falls from 76% in round 1 to 23% in round 10 (Section 4.3). As a result, maj@128 falls in 7 of the 12 self-voting settings (Figure 4b). Labels that do not come from the solver's vote avoid this: maj@128 rises in 17 of the 18 settings whose labels come from the proposer (Vocabulary Dropout) or an environment (AZR, SPIRAL), and SPICE's document labels keep it nearly unchanged, except on OctoThinker, where it falls by 19 points.

Only human-annotated labels correct substantially more errors. Keeping correct answers is not the same as correcting wrong ones. On initially not-correct questions, the percentage of correct responses rises by a median of 3–4 pp for every self-evolving label source, about half the 7.0 of human-annotated RL (Figure 4c). Correcting an error takes a correct label on a mathematics question the solver gets wrong, and only human annotation provides one for nearly every training question. Proposer-stated answers are often wrong after the first round (Section 4.3), and AZR and SPIRAL learn from code and games, whose rewards never check a mathematical answer, so their gains on mathematics come from transfer.

Summary. Self-evolving RL mostly teaches the solver to give answers it already knows more often and in a format the grader reads; it fixes few errors and rarely enlarges the set of questions it can solve. Labels from the proposer or an environment keep correct answers from eroding but fix no more errors than self-voting; only human-annotated labels fix substantially more.

4 Later Proposers Do Not Teach More

We now turn to our second question: does the proposer generate more useful training tasks as it evolves, as sustained self-improvement requires? We test this in three ways: a controlled study of the question source in R-Zero, a comparison of early and later proposers on a fixed solver in five methods (R-Zero, SPIRAL, AZR, Vocabulary Dropout, and SPICE), and a review of the generated tasks.

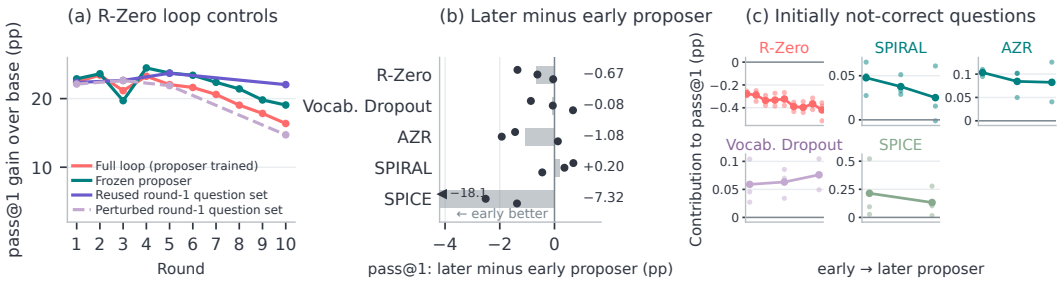

Figure 6: Proposer tests on Qwen3-8B. (a) R-Zero question sources. (b) Later minus early proposer. (c) Initially not-correct questions, early to later proposers (own scales). Dots in (b,c): three seeds.

4.1 A Controlled Study of the Question Source in R-Zero

We first test whether training the proposer helps within one method. We run 10 R-Zero rounds on Qwen3-8B and compare four question sources, with all other settings fixed: (i) the *full loop*, which updates the proposer every round; (ii) a *frozen proposer*, which stays at initialization but writes a new question set every round; (iii) the round-one question set, *reused* every round; and (iv) the round-one set *perturbed* by shifting each integer by a small random offset (Appendix D.1). In all four, the current solver labels the questions by majority vote, so the question source is the only difference.

Training the proposer does not help. As shown in Figure 6a, all four conditions yield nearly the same gain over the base model after round one, about 22 pp. By round ten, the full loop falls to +16.4 pp, below both the frozen proposer and the reused question set.

Early gains hold even with invalid questions. We sample 500 questions from each of the two round-one question sets and ask Claude Opus 5.5, blind to their source, whether each question is valid. Two independent passes both judge 30.2% of the original and 56.0% of the perturbed questions invalid (Figure 7a). Nevertheless, the two question sets yield the same round-one gain, +22.4 and +22.1 pp (Figure 7b). The perturbed set falls behind only later, reaching +14.7 pp at round ten. This fits Section 3: early gains reinforce answers the solver already has.

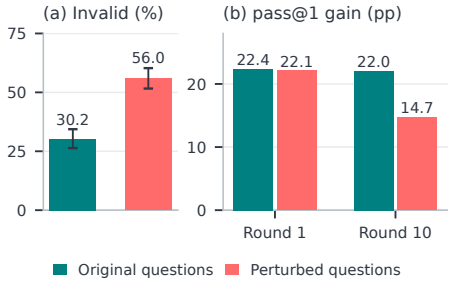

Figure 7: Original vs. perturbed round-one questions: (a) questions judged invalid (%; 95% interval); (b) pass@1 gain over the base model (pp).

4.2 Early vs. Later Proposers on a Fixed Solver

Because the solver and the proposer change together during training, we compare proposers on a fixed solver. For each of five methods, we train copies of one starting Qwen3-8B solver on tasks from an early and a later proposer, with matched budgets and three seeds, each used for both the early and the later proposer (see Appendix D.3 for details); a positive later-minus-early difference in pass@1 means the later proposer teaches more.

No method consistently benefits from a later proposer. In all five methods, later tasks teach no more than early ones (Figure 6b). For R-Zero, Vocabulary Dropout, AZR, and SPIRAL, the difference stays within about ± 1 pp, and only R-Zero's is consistent across runs, with the later proposer worse in all three paired runs. For SPICE, the later proposer is worse in all three runs, by 1.4 to 18.1 pp: its tasks list answer options without a question, and the correct option is always A, so the solver learns to answer "A" (Appendix D.3). Every Vocabulary Dropout task source helps the starting solver, but later ones add nothing.

No proposer teaches the solver to fix more errors. We split each change by question group, as in Section 3.2, with groups defined by the starting solver. For every task source of every method, the questions the starting solver answers incorrectly contribute at most about 0.2 pp (Figure 6c). With R-Zero tasks, their contribution is negative for every proposer round and drifts lower in later rounds, consistent with majority-vote labels reinforcing wrong leading answers (Section 5).

4.3 Quality of Generated Tasks across Rounds

To see why later proposers teach no more, we examine the generated tasks directly. Every task pairs a question with a *label*, the answer the solver is rewarded for matching. Where the label comes from depends on the method. In R-Zero, with or without a trained proposer, the solver answers each

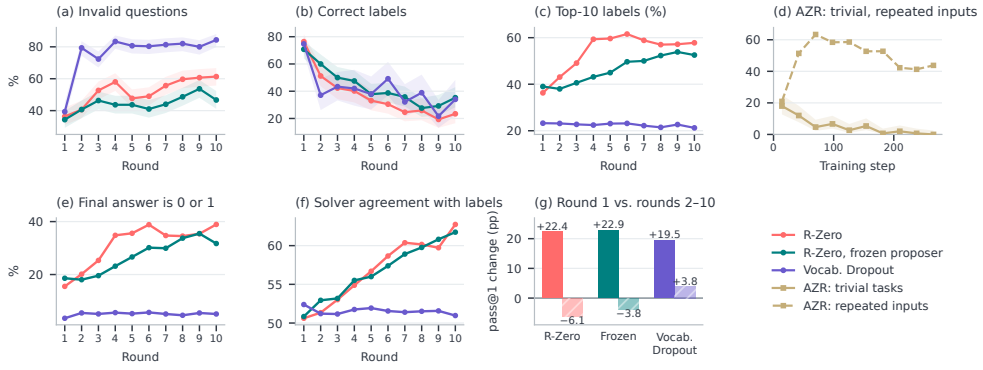

Figure 8: Generated tasks. (a–f) Qwen3-8B (AZR in 28-step windows; 95% bands); (g) pass@1 change in round 1 (solid) vs. rounds 2–10 (hatched).

question several times and its most common answer becomes the label. In Vocabulary Dropout, the proposer writes the answer together with the question. In AZR, the proposer writes a program and an input, and the label is the output recorded when the program ran. A task can teach the solver only if the question makes sense, its label is right, answering it needs the intended reasoning, and the round offers varied tasks, so we check four properties. *Validity:* a question is valid if its conditions can all hold at once and fix a single answer; one with contradictory conditions (e.g., a triangle with sides 1, 2 and 5) or a missing quantity is invalid. *Label correctness:* the reviewer solves each valid question and checks whether the label equals its answer, allowing equivalent forms; for AZR, it runs the program and compares the result with the recorded output. *Triviality:* a task is trivial if it can be solved without the computation it is meant to test; for AZR, an input that skips the program's main computation, such as an empty list. *Diversity:* we report the percentage of a round's questions whose label is one of its ten most common labels (for AZR, the percentage of tasks whose input repeats an earlier one). We use Claude Opus 5.5 as the reviewer, to rate 300 sampled questions per round in three Qwen3-8B math runs (R-Zero, R-Zero with the frozen proposer, and Vocabulary Dropout) and to execute 150 AZR output-prediction tasks from each of ten 28-step windows. The reviewer does not see which run or round a task comes from (details and prompts in Appendix D.2).

Self-voted math tasks get worse on every check. In R-Zero, invalid questions rise from 36% in round 1 to 61% in round 10, and correct labels fall from 76% to 23% (Figure 8a,b). The questions also become more uniform: the ten most common labels cover 58% of round-10 questions, up from 36% (panel c), and questions whose final answer is 0 or 1, a simple sign of trivial answers, rise from 16% to 39% (panel e), a shift seen in all five R-Zero models analyzed (Appendix D.2). The frozen proposer shows the same trends more slowly, so the decline comes from the solver, which selects and labels the questions; its agreement with the kept labels rises (panel f). Vocabulary Dropout, whose labels come from the proposer, keeps its labels diverse on all six models, but after round 1 about 80% of its questions are invalid.

Better tasks do not bring more learning. AZR's tasks get better on two checks: of 1,500 reviewed tasks, all but three stored outputs match execution, and trivial tasks fall from 18% in the first 28-step window to none in the last (panel d). Yet later AZR tasks teach a fixed solver no more (Section 4.2). In the math runs, the solver gains 19 to 23 pp in round one, while rounds 2–10 together change pass@1 by -6.1 pp for R-Zero, -3.8 with the frozen proposer, and $+3.8$ for Vocabulary Dropout (panel g).

Summary. Training the proposer does not yet supply more useful tasks. In round one, an untrained proposer, reused questions, or a mostly invalid question set teach the solver as much as the trained loop. Tasks from later proposers teach a fixed solver no more than tasks from early ones and correct no more errors, whether task quality falls over training, as in R-Zero, or rises, as in AZR.

5 Theoretical Analysis of Solver Updates

Sections 3 and 4 show that solver gains mainly reinforce answers the solver already gets right and that later proposers do not teach more. In this section, we explore why, by analyzing a single solver update on one training question in a simplified setting. The intuition is simple: an update can only strengthen answers the solver already gives, and the label decides which one. Let ν be the solver's distribution over final answers and $w(a)$ the probability that answer a becomes the training label. Human-annotated labels put all of w on the correct answer g; with self-voting, $w(a)$ is the probability

that a wins the vote, with random tie-breaking, and passes the agreement band of Eq. (2). We model the update for a sampled label $B \sim w$ as the KL-regularized optimum and write ν_B^+ for the solver's answer distribution after it: $\nu_B^+(a) \propto \nu(a)\, e^{\lambda \mathbf{1}\{a=B\}}$, which multiplies the probability of B by e^λ and renormalizes; $\lambda > 0$ is the step size.

Proposition 1 (The label source decides which answer gains)**.** *For any label source w and any two answers a, b with $\nu(a), \nu(b) > 0$,*

$$\mathbb{E}_B\left[\log \frac{\nu_B^+(a)}{\nu_B^+(b)} - \log \frac{\nu(a)}{\nu(b)}\right] = \lambda\,\big[w(a) - w(b)\big]. \tag{5}$$

Under majority voting, the solver's most likely answer has the largest w, so in expectation it gains log-odds against every other answer, right or wrong.

With human-annotated labels, g instead gains λ in log-odds against every wrong answer. To see how accuracy $p = \nu(g)$ changes, expand the expected update to first order in λ:

$$\mathbb{E}_B[\nu_B^+(g) - p] = \lambda p(1-p)D + O(\lambda^2), \qquad D = w(g) - \sum_{b \neq g} \frac{\nu(b)}{1-p}\, w(b). \tag{6}$$

The factor $p(1-p)$ does not depend on the label source: even with correct labels, a question the solver rarely answers correctly gains little per update. The label source enters only through D, the probability of a correct label minus the solver-weighted probability of wrong labels. Human-annotated labels give $D = 1$ on every question. Majority-vote labels give a large D where the solver already favors g, and can give $D < 0$ where it favors a wrong answer.

Proposition 2 (Proposers reweight questions)**.** *Fix the solver and its labeling and acceptance rule, and let a proposer sample questions from μ_ϕ. A question q is kept with probability $\alpha(q)$, so training questions follow $\tilde{\mu}_\phi(q) \propto \mu_\phi(q)\alpha(q)$ and the expected first-order gain is $\lambda\, \mathbb{E}_{q\sim\tilde{\mu}_\phi}[p_q(1-p_q)D_q]$, where p_q and D_q do not depend on ϕ. Hence no proposer changes the sign of a question's update.*

A better proposer can favor questions where D_q is large but cannot turn a negative D_q positive, and the result gives no ordering between early and later proposers (Section 4.2). Even a frozen proposer's question quality can fall, since which questions are kept depends on the solver's agreement κ_t.

The measured gains agree with the model. The model predicts three patterns, and the data show each. First, by the factor $p(1-p)$, questions the solver rarely answers correctly should gain little under any label source: the base models answer initially not-correct questions correctly in only 2–4% of samples, and every label source draws 80–98% of its gain from initially correct questions (Figure 3). Second, since correct labels give $D = 1$, human-annotated RL should gain more on these questions, and it gains more than twice as much (Section 3.2). Third, self-voted labels should strengthen a wrong plurality answer: where that answer is a number we can track, R-Zero raises its probability by 16 pp on average, about twice as much as any other method in Appendix Table 14. These are consistency checks rather than tests: the model treats each question in isolation and simplifies the GRPO update.

6 Conclusion and Limitations

Ideally, a proposer and a solver that optimize each other would start an endless cycle of improvement. We asked what actually evolves in this cycle, for the solver and for the proposer. Across six methods and six base models, solver gains mainly reinforce answers the base model already gets right. Labels from code execution or games protect these answers but correct far fewer errors than human-annotated labels. The proposer's tasks change during training but do not become more useful: later proposers do not teach a fixed solver more than early ones. Our analysis of one update agrees: even correct labels barely raise answers the solver rarely gives, and majority-vote labels strengthen whatever answer the solver already favors. These results suggest evaluating self-evolving RL not only by pass@1, but also by how many errors the solver corrects and how much proposer updates add to what a fixed solver learns. **Taken together, current gains mainly reflect sharpening, and self-evolving RL has so far only partially achieved sustained recursive self-improvement.**

Our study has limitations. It is bounded by compute: some runs take more than a week on eight GPUs, so we cover six 4–8B base models and only mathematical reasoning. Larger models (14B and above) and other domains, such as code or agentic tasks, may behave differently; studying them would further test these conclusions. Finally, some methods have no official code, so we reimplement them from their papers as faithfully as possible, and newer methods in this fast-moving field may address some of these limitations.

ETHICS STATEMENT

This work studies published training methods on public mathematical benchmarks. It involves no human subjects or personal data, and the generated training tasks we review contain only synthetic math problems and programs. Our findings bear on the safe use of self-improving systems: labels a model produces for itself can reinforce its own wrong answers, so a rise in accuracy should not be taken as evidence that such a system corrects its errors.

REPRODUCIBILITY STATEMENT

Section 2.2 and Appendix D.1, D.2, and D.3 give the models, training schedules, evaluation protocol, and audit procedures. We will release the training and evaluation code, all 350 trained checkpoints with their sampled responses, the per-question grades, and the audit samples with their judgments.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods and write code, to help analyze results, to make figures, to search the literature, and to edit the paper. Generative AI is also part of our measurements: Claude Haiku 4.5 rescores responses and escalates disagreements to Claude Opus 5 (Section 2.2 and Appendix E.1.2), and Claude Opus 5.5 audits the generated training tasks (Sections 4.1 and 4.3 and Appendix D.2) and checks the reasoning of correct AIME responses (Appendix B.3). We have not used generative AI tools for synthetic dataset generation: the training tasks we analyze are generated by the self-evolving methods under study. We have reviewed all AI-assisted work: every reported number is computed by scripts from the saved evaluation outputs and checked against those records, and every LLM-judged result is reported next to the rule-based grader. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, et al. SmolLM2: When smol goes big – data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*, 2025.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Jacob Dineen, Aswin RRV, Zhikun Xu, and Ben Zhou. Vocabulary dropout for curriculum diversity in LLM co-evolution. In *Conference on Language Modeling*, 2026. arXiv:2604.03472.

Kefan Dong and Tengyu Ma. STP: Self-play LLM theorem provers with iterative conjecturing and proving. In *International Conference on Machine Learning (ICML)*, 2025. arXiv:2502.00212.

Marius Dragoi, Ioana Pintilie, Florin Gogianu, and Florin Brad. Beyond pass@k: Breadth-depth metrics for reasoning boundaries. *arXiv preprint arXiv:2510.08325*, 2025. NeurIPS 2025 Workshop on Foundations of Reasoning in Language Models.

Lishui Fan, Mouxiang Chen, Tingwei Zhu, Kui Liu, Xin Xia, Shanping Li, and Zhongxin Liu. ZeroCoder: Can LLMs improve code generation without ground-truth supervision? *arXiv preprint arXiv:2604.07864*, 2026.

Irving John Good. Speculations concerning the first ultraintelligent machine. *Advances in Computers*, 6:31–88, 1965.

Bingxiang He, Yuxin Zuo, Zeyuan Liu, Shangziqi Zhao, Zixuan Fu, Junlin Yang, Cheng Qian, Kaiyan Zhang, Yuchen Fan, Ganqu Cui, Xiusi Chen, Youbang Sun, Xingtai Lv, Xuekai Zhu, Li Sheng, Ran Li, Huan-ang Gao, Yuchen Zhang, Bowen Zhou, Zhiyuan Liu, and Ning Ding. How far can unsupervised RLVR scale LLM training? *arXiv preprint arXiv:2603.08660*, 2026.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

hiyouga. MathRuler. `https://github.com/hiyouga/MathRuler`, 2025.

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-Zero: Self-evolving reasoning LLM from zero data. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2508.05004.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35, 2022.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, et al. DataComp-LM: In search of the next generation of training sets for language models. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024.

Hongxiang Lin, Zhirui Kuai, Erpeng Xue, and Lei Wang. Detecting and mitigating the correct-answer extinction window in test-time reinforcement learning with majority voting. *arXiv preprint arXiv:2605.19444*, 2026.

Jianzhe Lin. Self-improvement can self-regress: The rise-and-collapse failure mode of LLM self-training. *arXiv preprint arXiv:2606.21090*, 2026.

Bo Liu, Chuanyang Jin, Seungone Kim, Weizhe Yuan, Wenting Zhao, Ilia Kulikov, Xian Li, Sainbayar Sukhbaatar, Jack Lanchantin, and Jason Weston. SPICE: Self-play in corpus environments improves reasoning. *arXiv preprint arXiv:2510.24684*, 2025.

Bo Liu, Leon Guertler, Simon Yu, Zichen Liu, Penghui Qi, Daniel Balcells, Mickel Liu, Cheston Tan, Weiyan Shi, Min Lin, Wee Sun Lee, and Natasha Jaques. SPIRAL: Self-play on zero-sum games incentivizes reasoning via multi-agent multi-turn reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2506.24119.

MAA. American Mathematics Competitions (AMC) 2023. `https://maa.org/student-programs/amc/`, 2023.

MAA. American Invitational Mathematics Examination (AIME) 2024. `https://maa.org/math-competitions/american-invitational-mathematics-examination-aime`, 2024.

MAA. American Invitational Mathematics Examination (AIME) 2025. `https://maa.org/math-competitions/american-invitational-mathematics-examination-aime`, 2025.

Kohsei Matsutani, Shota Takashiro, Gouki Minegishi, Takeshi Kojima, Yusuke Iwasawa, and Yutaka Matsuo. RL squeezes, SFT expands: A comparative study of reasoning LLMs. *arXiv preprint arXiv:2509.21128*, 2025. ICLR 2026.

Mihir Prabhudesai, Lili Chen, Alex Ippoliti, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. Maximizing confidence alone improves reasoning. *arXiv preprint arXiv:2505.22660*, 2025.

Jürgen Schmidhuber. Gödel machines: Fully self-referential optimal universal self-improvers. In *Artificial General Intelligence*, pp. 199–226. Springer, 2007.

Sheikh Shafayat, Fahim Tajwar, Ruslan Salakhutdinov, Jeff Schneider, and Andrea Zanette. Can large reasoning models self-train? *arXiv preprint arXiv:2505.21444*, 2025.

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. *arXiv preprint arXiv:2402.03300*, 2024.

Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base LLMs. In *International Conference on Learning Representations*, 2026.

Peng Xia, Kaide Zeng, Jiaqi Liu, Can Qin, Fang Wu, Yiyang Zhou, Caiming Xiong, and Huaxiu Yao. Agent0: Unleashing self-evolving agents from zero data via tool-integrated reasoning. In *Conference on Language Modeling (COLM)*, 2026. arXiv:2511.16043.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/537d5aa768c2d534016a4d06f87bc8fb-Abstract-Conference.html.

Zhenrui Yue, Kartikeya Upasani, Xianjun Yang, Suyu Ge, Shaoliang Nie, Yuning Mao, Zhe Liu, and Dong Wang. Dr. zero: Self-evolving search agents without training data. In *Conference on Language Modeling (COLM)*, 2026. arXiv:2601.07055.

Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. Self-taught optimizer (STOP): Recursively self-improving code generation. In *Conference on Language Modeling (COLM)*, 2024.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Gödel machine: Open-ended evolution of self-improving agents. In *International Conference on Learning Representations (ICLR)*, 2026.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025a. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/9837dc00ff67d176373268ed48042d49-Abstract-Conference.html.

Rosie Zhao, Alexandru Meterez, Sham Kakade, Cengiz Pehlevan, Samy Jelassi, and Eran Malach. Echo chamber: RL post-training amplifies behaviors learned in pretraining. *arXiv preprint arXiv:2504.07912*, 2025b. COLM 2025.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. Learning to reason without external rewards. *arXiv preprint arXiv:2505.19590*, 2025c.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2023.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, Biqing Qi, Youbang Sun, Zhiyuan Ma, Lifan Yuan, Ning Ding, and Bowen Zhou. TTRL: Test-time reinforcement learning. In *Advances in Neural Information Processing Systems*, 2025. arXiv:2504.16084.

APPENDIX CONTENTS

A Related Work **15**

B Metrics and answer accounting **15**
B.1 Primary answer accounting and semantic sensitivity 15
B.2 Peak, vote-reference, and endpoint diagnostics 19
B.3 Checking the reasoning of correct responses on AIME 21

C Training trajectories **22**
C.1 Cross-model trajectories 22

D Loop interventions and supervision **23**
D.1 Changing the question source inside the R-Zero loop 23
D.2 Round-by-round task audit 23
D.3 Proposer comparison with a fixed starting solver 25
D.4 Errors in generated training tasks 27
D.5 Original R-Zero evaluation protocol 28

E Evaluation protocol and reproducibility **28**
E.1 Evaluation setting and experiment taxonomy 28
E.2 Paper–code configuration audit 30

F KL-proximal analysis of label-directed reinforcement **30**
F.1 Full statements of the main-text results 31
F.2 Accepted-experience distribution 32
F.3 Setup 32
F.4 Self-vote selection and update 33
F.5 Label source and reinforcement direction 34
F.6 Initial-answer reinforcement and correction at selected peaks 35
F.7 Pass@k and correct-answer probability thresholds 35
F.8 Proofs and finite-sample qualifications 37

G Further Tests **38**

A Related Work

Self-evolving RL. Recent methods train language models on tasks they generate themselves, with self-voted labels (Huang et al., 2026), executable tasks (Zhao et al., 2025a), document grounding (Liu et al., 2025), self-play games (Liu et al., 2026), tool use (Xia et al., 2026), or diversified curricula (Dineen et al., 2026), and extend to search, code, and theorem proving (Yue et al., 2026; Fan et al., 2026; Dong & Ma, 2025). These works mainly report accuracy gains; we ask what the gains consist of and whether the proposer contributes to them.

Learning without external labels. Test-time and unsupervised RL train on majority-vote or confidence rewards (Zuo et al., 2025; Shafayat et al., 2025; Zhao et al., 2025c; Prabhudesai et al., 2025), which can amplify a wrong consensus and collapse (He et al., 2026; Lin et al., 2026; Lin, 2026); R-Zero itself reports declining pseudo-label accuracy (Huang et al., 2026). We measure this effect across six self-evolving methods and relate it to the proposer.

What RL with verifiable rewards learns. RLVR raises pass@1 but rarely expands what a base model can solve at large k (Yue et al., 2025), and mostly amplifies behaviors present after pretraining (Zhao et al., 2025b; Matsutani et al., 2025). We ask the same question for self-evolving RL, where the tasks and labels are also generated by the model.

B Metrics and answer accounting

B.1 Primary answer accounting and semantic sensitivity

The primary analysis uses the saved rule-based judgments on $N = 2{,}866$ questions with $n = 128$ responses per checkpoint. The evaluation code uses `extract_boxed_content` for answer extraction and `grade_answer` for reference-based grading. Each response is either correct, parsed but incorrect, or unparsed. Their empirical frequencies $\hat{p}_t(q)$, $\hat{w}_t(q)$, and $\hat{u}_t(q)$ sum to one. Consequently,

$$\hat{p}_t(q) - \hat{p}_0(q) = \hat{u}_0(q) - \hat{u}_t(q) + \hat{w}_0(q) - \hat{w}_t(q).$$

This identity describes changes in response percentages, not a matching of individual samples across checkpoints. An unparsed response need not lack a meaningful answer.

The rule-based grader clusters parsed answers using its saved equivalence judgments. The leading cluster (the plurality answer) has the largest count; ties retain the first cluster encountered in sample order. Ties occur in 6.1% of question–setting pairs and change the correctness of the plurality answer in 1.1%. We use the saved correctness label of that cluster. Unparsed responses remain in the denominator and do not compete for the lead. The six question classes are:

Class	Initial plurality answer	Peak plurality answer
Stayed correct	correct	correct
Became correct	incorrect	correct
Became wrong	correct	incorrect
Same W	incorrect	same incorrect answer
Different W	incorrect	different incorrect answer
Abstention	all 128 responses unparsed at either checkpoint	

The main text groups questions only by their initial plurality answer. Initially correct questions are the first and third classes, plus the rare abstentions whose initial plurality answer is correct; initially not-correct questions are all others. The six classes refine these two groups by the peak outcome; abstentions contribute at most 0.01 pp in any setting. Incorrect leaders match across checkpoints when their representative renderings coincide after removing whitespace, dollar signs, and the LaTeX size commands `\left` and `\right`. This records answer identity under the stated normalization, not semantic novelty.

For class s, the primary contribution is

$$G_s^M(t) = \frac{1}{N} \sum_{q:S_t(q)=s} [\hat{p}_t^M(q) - \hat{p}_0^M(q)], \qquad \widehat{\Delta \mathrm{pass@1}_t}^M = \sum_s G_s^M(t).$$

Every question appears once. "Net" denotes this sum, not a seventh class. A transition label does not fix the sign of its probability contribution: a correct answer can lose the lead while its own probability increases. Class-internal improvements and losses can also cancel. Corrections and losses of leading answers are reported separately. Their count difference omits transitions involving all-unparsed outcomes, so it need not equal the total change in the number of correct-leading questions.

What the LLM judge adds. The LLM judge identifies the response's final answer when one is present and checks its equivalence to the reference. A *committed answer* is an answer the response presents as its solution, rather than an intermediate calculation or a candidate it later rejects. The judge does not execute code or finish calculations. Its *abstention* label means that it could not identify a complete final answer; recorded deterministic refusal fallbacks also receive this label. Unresolved judgments are excluded from completed scoring records. This differs from the rule-based grader's failure to parse an answer: a response may state a complete answer without putting it in the required boxed format. We retain the pass@1-selected checkpoint and the primary question partition when replacing response correctness with the semantic labels to obtain G_s^J. This follow-up studies semantic-scored gains within the primary question groups.

Let $r_t(q)$ be the fraction of all responses on q that the rule-based grader leaves unparsed but the LLM judge labels correct. Define

$$F_s(t) = \frac{1}{N} \sum_{q:S_t(q)=s} [r_0(q) - r_t(q)], \qquad D_s(t) = G_s^M(t) - G_s^J(t) - F_s(t).$$

The scoring decomposition is therefore

$$G_s^M = G_s^J + F_s + D_s. \tag{7}$$

The extraction term F_s measures how changes in missed correct answers affect the primary gain; D_s contains the net change in other correctness disagreements. A positive F_s means fewer semantically correct answers are missed at the peak. These are scoring sensitivity terms, not causal effects of format training. They are measured within every question group, including stayed correct, rather than assigned to the all-unparsed question class. Restricted to the questions whose plurality answer becomes correct at the peak, F is a median 9% of G^M across all 42 settings, so the few corrected errors are mostly confirmed by the judge. In one setting, SPICE on OctoThinker-8B, F exceeds the whole gain (361% of G^M): its MathRuler gain is $+1.7$ pp while its judged pass@1 falls by 4.2, so Figure 4a leaves this value off its axis.

For any sample budget $1 \le k \le n$, the unbiased pass@k estimator is $h_k(c) = 1 - \binom{n-c}{k} / \binom{n}{k}$. Replacing $\hat{p}$ by $h_k(c)$ gives the corresponding additive accounting. At $k = n = 128$, pass@128 change is the number of questions newly solved minus those no longer solved within 128 attempts, divided by N.

Peak selection. All 350 registered trained checkpoints have complete rule-based scores. We select the maximum pooled pass@1 in each of the 42 settings, breaking ties by the earliest coordinate. Initialization is the reference and is excluded from the trained candidates. The selected checkpoint is held fixed in the LLM-judge comparison; semantic labels are available for all selected checkpoints and initializations. This is a descriptive maximum over the registered grid, not an independently validated stopping rule. For the selected peak $t^\star$, the gain is $100(\widehat{\text{pass@1}}_{t^\star} - \widehat{\text{pass@1}}_0)$ pp. The endpoint is the last registered training coordinate; endpoint gains replace $t^\star$ with that coordinate. Post-peak change is the endpoint score minus the peak score. Pass@128 gains use the corresponding pass@128 scores.

The main-text percentage divides the contribution of initially correct questions by the net pass@1 gain; it ranges from 80% to 98% (median 95.9%). In the six-class accounting, the percentage of the positive contributions that comes from stayed-correct questions, $G^M_{\text{C}\to\text{C}} / \sum_s \max(G_s^M, 0)$, ranges from 90% to 98% (median 95.9%). Table 2 reports the contributions; Table 3 gives class sizes and within-class movements. All displayed contributions are in pp over all 2,866 questions, rather than percentages within the class.

Table 2: Pass@1 gain contributions at MathRuler-selected peaks (pp).

Model	Recipe	S/R	Peak	C→C	W→C	C→W	Same W	Other W	Abstain	Net
Qwen3-4B	R-Zero	10/10	r2	+24.01	+0.39	+0.06	+0.14	+0.11	+0.00	+24.72
Qwen3-8B	R-Zero	10/10	r2	+22.99	+0.51	+0.07	+0.17	+0.12	+0.00	+23.86
Qwen2.5-7B	R-Zero	10/10	r2	+39.25	+1.15	+0.16	+0.39	+0.20	+0.00	+41.15
OLMo-3-7B	R-Zero	10/10	r6	+19.60	+0.56	-0.78	+0.17	+0.23	+0.00	+19.78
OctoThinker-8B	R-Zero	10/10	r3	+28.42	+1.02	+0.32	+0.22	+0.29	+0.00	+30.26
Granite-4.1-8B	R-Zero	10/10	r1	+35.28	+0.41	+0.10	+0.17	+0.11	-0.00	+36.07
Qwen3-4B	SPICE	7/7	s100	+27.96	+0.75	-0.02	+0.04	+0.08	+0.00	+28.81
Qwen3-8B	SPICE	7/7	s100	+26.66	+1.20	-0.06	+0.05	+0.10	+0.00	+27.96
Qwen2.5-7B	SPICE	7/7	s200	+41.25	+1.68	+0.18	+0.36	+0.31	+0.00	+43.78
OLMo-3-7B	SPICE	2/2	s100	+12.79	+0.14	-0.02	+0.06	+0.05	+0.00	+13.01

Model	Recipe	S/R	Peak	C→C	W→C	C→W	Same W	Other W	Abstain	Net
OctoThinker-8B	SPICE	7/7	s200	+3.23	+0.34	-1.86	-0.02	+0.03	+0.00	+1.72
Granite-4.1-8B	SPICE	7/7	s200	+18.76	+0.60	-0.27	+0.07	+0.11	+0.00	+19.27
Qwen3-4B	SPIRAL	10/10	s400	+23.96	+0.59	+0.03	+0.17	+0.16	+0.00	+24.91
Qwen3-8B	SPIRAL	10/10	s400	+25.50	+0.70	+0.04	+0.16	+0.12	+0.00	+26.51
Qwen2.5-7B	SPIRAL	10/10	s40	+30.54	+0.51	+0.11	+0.44	+0.13	+0.00	+31.72
OLMo-3-7B	SPIRAL	10/10	s360	+10.94	+0.19	+0.01	+0.16	+0.08	+0.00	+11.37
OctoThinker-8B	SPIRAL	10/10	s200	+26.48	+0.96	+0.11	+0.42	+0.32	+0.00	+28.29
Granite-4.1-8B	SPIRAL	10/10	s320	+11.53	+0.12	-0.02	+0.06	+0.04	+0.00	+11.73
Qwen3-4B	AZR	3/3	s390	+16.41	+0.50	+0.02	+0.20	+0.09	+0.00	+17.22
Qwen3-8B	AZR	3/3	s390	+7.92	+0.33	-0.01	+0.15	+0.06	+0.00	+8.45
Qwen2.5-7B	AZR	3/3	s260	+12.88	+0.60	+0.04	+0.24	+0.10	+0.00	+13.86
OLMo-3-7B	AZR	3/3	s390	+3.38	+0.12	-0.14	-0.02	-0.02	-0.01	+3.30
OctoThinker-8B	AZR	3/3	s260	+19.07	+0.78	+0.02	+0.08	+0.19	+0.00	+20.13
Granite-4.1-8B	AZR	3/3	s390	+13.95	+0.23	-0.03	+0.07	+0.03	+0.00	+14.24
Qwen3-4B	Agent0	10/10	s100	+19.81	+0.27	+0.06	+0.17	+0.09	+0.00	+20.39
Qwen3-8B	Agent0	10/10	s90	+20.45	+0.29	+0.07	+0.24	+0.11	+0.00	+21.17
Qwen2.5-7B	Agent0	10/10	s60	+22.48	+0.42	+0.10	+0.32	+0.14	+0.00	+23.46
OLMo-3-7B	Agent0	10/10	s100	+10.33	+0.41	-0.07	+0.18	+0.12	+0.00	+10.97
OctoThinker-8B	Agent0	10/10	s70	+21.75	+0.52	+0.04	+0.18	+0.12	+0.00	+22.62
Granite-4.1-8B	Agent0	10/10	s80	+36.52	+0.74	+0.08	+0.12	+0.12	+0.00	+37.58
Qwen3-4B	Vocab. Dropout	10/10	r8	+23.77	+0.43	+0.02	+0.18	+0.13	+0.00	+24.54
Qwen3-8B	Vocab. Dropout	10/10	r10	+22.46	+0.48	+0.05	+0.20	+0.12	+0.00	+23.32
Qwen2.5-7B	Vocab. Dropout	10/10	r10	+37.22	+0.87	+0.13	+0.53	+0.22	+0.00	+38.98
OLMo-3-7B	Vocab. Dropout	5/5	r5	+17.76	+0.25	+0.03	+0.15	+0.05	+0.00	+18.24
OctoThinker-8B	Vocab. Dropout	10/10	r9	+25.99	+0.81	+0.10	+0.40	+0.28	+0.00	+27.58
Granite-4.1-8B	Vocab. Dropout	10/10	r4	+34.91	+0.72	+0.09	+0.14	+0.10	+0.00	+35.97
Qwen3-4B	Human-annotated RL	10/10	s30	+26.70	+1.62	+0.00	+0.22	+0.15	+0.00	+28.70
Qwen3-8B	Human-annotated RL	10/10	s150	+26.64	+2.01	-0.06	+0.14	+0.12	+0.00	+28.85
Qwen2.5-7B	Human-annotated RL	10/10	s30	+39.01	+1.16	+0.13	+0.42	+0.25	+0.00	+40.97
OLMo-3-7B	Human-annotated RL	10/10	s60	+26.85	+0.86	+0.08	+0.17	+0.17	+0.00	+28.12
OctoThinker-8B	Human-annotated RL	10/10	s150	+29.02	+1.29	-0.16	+0.31	+0.33	+0.00	+30.78
Granite-4.1-8B	Human-annotated RL	10/10	s105	+37.08	+1.47	+0.03	+0.20	+0.27	+0.00	+39.05

Table 3: Question counts and within-class correct-probability changes at MathRuler-selected peaks.

Model	Recipe	C→C	W→C	C→W	Same W	Other W	Abstain
Qwen3-4B	R-Zero	2005 / +34.3	53 / +21.3	78 / +2.2	423 / +1.0	307 / +1.1	0 / —
Qwen3-8B	R-Zero	2053 / +32.1	58 / +25.2	72 / +2.6	380 / +1.3	303 / +1.2	0 / —
Qwen2.5-7B	R-Zero	1928 / +58.3	101 / +32.5	55 / +8.4	446 / +2.5	336 / +1.7	0 / —
OLMo-3-7B	R-Zero	1843 / +30.5	52 / +30.9	279 / -8.1	235 / +2.0	457 / +1.4	0 / —
OctoThinker-8B	R-Zero	1650 / +49.4	74 / +39.5	133 / +6.9	400 / +1.6	609 / +1.3	0 / —
Granite-4.1-8B	R-Zero	2253 / +44.9	67 / +17.3	23 / +12.5	196 / +2.5	319 / +1.0	8 / -0.2
Qwen3-4B	SPICE	2000 / +40.1	54 / +39.7	83 / -0.5	385 / +0.3	344 / +0.7	0 / —
Qwen3-8B	SPICE	2043 / +37.4	95 / +36.3	82 / -2.2	304 / +0.5	342 / +0.9	0 / —
Qwen2.5-7B	SPICE	1878 / +63.0	100 / +48.2	105 / +4.9	375 / +2.8	408 / +2.2	0 / —
OLMo-3-7B	SPICE	2063 / +17.8	69 / +5.6	59 / -1.2	254 / +0.6	421 / +0.4	0 / —
OctoThinker-8B	SPICE	1203 / +7.7	38 / +25.6	580 / -9.2	183 / -0.3	862 / +0.1	0 / —
Granite-4.1-8B	SPICE	2189 / +24.6	81 / +21.2	87 / -8.8	170 / +1.1	339 / +0.9	0 / —
Qwen3-4B	SPIRAL	2037 / +33.7	76 / +22.2	46 / +1.9	398 / +1.2	309 / +1.5	0 / —
Qwen3-8B	SPIRAL	2064 / +35.4	75 / +26.6	61 / +1.8	361 / +1.2	305 / +1.1	0 / —
Qwen2.5-7B	SPIRAL	1931 / +45.3	94 / +15.5	52 / +5.9	448 / +2.8	341 / +1.1	0 / —
OLMo-3-7B	SPIRAL	2063 / +15.2	70 / +7.7	59 / +0.7	282 / +1.6	392 / +0.6	0 / —
OctoThinker-8B	SPIRAL	1737 / +43.7	131 / +21.0	46 / +6.6	399 / +3.1	553 / +1.6	0 / —
Granite-4.1-8B	SPIRAL	2242 / +14.7	39 / +8.9	34 / -1.6	265 / +0.6	286 / +0.4	0 / —
Qwen3-4B	AZR	2050 / +22.9	84 / +17.0	33 / +1.6	417 / +1.4	282 / +0.9	0 / —
Qwen3-8B	AZR	2092 / +10.9	83 / +11.4	33 / -0.5	361 / +1.2	297 / +0.5	0 / —
Qwen2.5-7B	AZR	1947 / +19.0	124 / +13.9	36 / +3.2	436 / +1.6	323 / +0.9	0 / —
OLMo-3-7B	AZR	2065 / +4.7	93 / +3.6	53 / -7.7	212 / -0.3	388 / -0.1	55 / -0.6
OctoThinker-8B	AZR	1665 / +32.8	87 / +25.7	118 / +0.4	328 / +0.7	668 / +0.8	0 / —
Granite-4.1-8B	AZR	2234 / +17.9	51 / +12.9	42 / -2.0	229 / +0.8	310 / +0.2	0 / —
Qwen3-4B	Agent0	2029 / +28.0	47 / +16.6	54 / +3.2	441 / +1.1	295 / +0.9	0 / —
Qwen3-8B	Agent0	2073 / +28.3	42 / +20.1	52 / +4.1	417 / +1.6	282 / +1.1	0 / —
Qwen2.5-7B	Agent0	1927 / +33.4	85 / +14.2	56 / +5.4	471 / +1.9	327 / +1.2	0 / —
OLMo-3-7B	Agent0	2022 / +14.6	78 / +15.2	100 / -1.9	264 / +1.9	402 / +0.8	0 / —
OctoThinker-8B	Agent0	1702 / +36.6	107 / +14.0	81 / +1.3	387 / +1.4	589 / +0.6	0 / —
Granite-4.1-8B	Agent0	2248 / +46.6	82 / +25.7	28 / +8.5	191 / +1.9	311 / +1.1	6 / +0.0
Qwen3-4B	Vocab. Dropout	2041 / +33.4	68 / +18.3	42 / +1.6	433 / +1.2	282 / +1.4	0 / —
Qwen3-8B	Vocab. Dropout	2074 / +31.0	64 / +21.6	51 / +2.8	378 / +1.6	299 / +1.2	0 / —
Qwen2.5-7B	Vocab. Dropout	1932 / +55.2	106 / +23.5	51 / +7.4	429 / +3.6	348 / +1.8	0 / —
OLMo-3-7B	Vocab. Dropout	2085 / +24.4	91 / +8.0	37 / +2.6	269 / +1.6	384 / +0.3	0 / —
OctoThinker-8B	Vocab. Dropout	1726 / +43.1	115 / +20.2	57 / +5.2	430 / +2.6	538 / +1.5	0 / —
Granite-4.1-8B	Vocab. Dropout	2244 / +44.6	81 / +25.4	32 / +8.5	188 / +2.2	321 / +0.9	0 / —
Qwen3-4B	Human-annotated RL	2039 / +37.5	161 / +28.9	44 / +0.2	265 / +2.4	357 / +1.2	0 / —
Qwen3-8B	Human-annotated RL	2096 / +36.4	166 / +34.7	29 / -5.9	247 / +1.6	328 / +1.1	0 / —
Qwen2.5-7B	Human-annotated RL	1944 / +57.5	113 / +29.5	39 / +9.4	418 / +2.9	352 / +2.0	0 / —

Model	Recipe	C→C	W→C	C→W	Same W	Other W	Abstain
OLMo-3-7B	Human-annotated RL	2086 / +36.9	115 / +21.4	36 / +6.1	247 / +2.0	382 / +1.3	0 / —
OctoThinker-8B	Human-annotated RL	1675 / +49.7	119 / +31.1	108 / -4.4	330 / +2.7	634 / +1.5	0 / —
Granite-4.1-8B	Human-annotated RL	2230 / +47.6	94 / +44.8	46 / +1.8	167 / +3.4	329 / +2.4	0 / —

Each entry is a question count / mean change in correct-answer probability (pp). Counts sum to 2,866 in each row.

Table 4: Gain sensitivity at fixed MathRuler-selected peaks (pp).

Model	Recipe	Peak	G^M	G^J	F	D	$A_0 - A_t$
Qwen3-4B	R-Zero	r2	+24.72	+17.42	+7.69	-0.39	+4.67
Qwen3-8B	R-Zero	r2	+23.86	+20.54	+4.53	-1.21	+1.20
Qwen2.5-7B	R-Zero	r2	+41.15	+23.92	+18.84	-1.61	+4.17
OLMo-3-7B	R-Zero	r6	+19.78	+7.16	+13.42	-0.81	+23.65
OctoThinker-8B	R-Zero	r3	+30.26	+24.40	+6.38	-0.51	+15.90
Granite-4.1-8B	R-Zero	r1	+36.07	+28.64	+9.37	-1.93	+16.50
Qwen3-4B	SPICE	s100	+28.81	+22.07	+7.76	-1.03	+4.63
Qwen3-8B	SPICE	s100	+27.96	+25.50	+4.66	-2.21	+0.87
Qwen2.5-7B	SPICE	s200	+43.78	+26.76	+18.94	-1.92	+5.05
OLMo-3-7B	SPICE	s100	+13.01	+3.73	+9.59	-0.32	+5.89
OctoThinker-8B	SPICE	s200	+1.72	-4.16	+6.19	-0.31	+7.14
Granite-4.1-8B	SPICE	s200	+19.27	+12.94	+7.09	-0.76	+18.83
Qwen3-4B	SPIRAL	s400	+24.91	+18.06	+7.77	-0.92	+4.21
Qwen3-8B	SPIRAL	s400	+26.51	+23.70	+4.60	-1.80	+0.56
Qwen2.5-7B	SPIRAL	s40	+31.72	+14.36	+18.41	-1.05	+4.77
OLMo-3-7B	SPIRAL	s360	+11.37	+2.09	+9.67	-0.39	+11.22
OctoThinker-8B	SPIRAL	s200	+28.29	+22.66	+6.36	-0.73	+15.11
Granite-4.1-8B	SPIRAL	s320	+11.73	+10.12	+2.19	-0.59	+9.83
Qwen3-4B	AZR	s390	+17.22	+12.72	+5.38	-0.88	+3.14
Qwen3-8B	AZR	s390	+8.45	+6.57	+2.41	-0.53	+0.56
Qwen2.5-7B	AZR	s260	+13.86	+4.35	+10.16	-0.66	+4.08
OLMo-3-7B	AZR	s390	+3.30	+2.35	+1.48	-0.54	-1.38
OctoThinker-8B	AZR	s260	+20.13	+16.46	+3.97	-0.30	+7.05
Granite-4.1-8B	AZR	s390	+14.24	+11.65	+3.85	-1.26	+11.17
Qwen3-4B	Agent0	s100	+20.39	+14.18	+6.75	-0.54	+4.42
Qwen3-8B	Agent0	s90	+21.17	+18.48	+3.94	-1.26	+1.22
Qwen2.5-7B	Agent0	s60	+23.46	+11.26	+13.05	-0.85	+4.37
OLMo-3-7B	Agent0	s100	+10.97	+7.02	+4.37	-0.42	+16.80
OctoThinker-8B	Agent0	s70	+22.62	+17.40	+5.79	-0.57	+12.14
Granite-4.1-8B	Agent0	s80	+37.58	+30.56	+9.73	-2.71	+19.04
Qwen3-4B	Vocab. Dropout	r8	+24.54	+17.22	+7.70	-0.37	+4.49
Qwen3-8B	Vocab. Dropout	r10	+23.32	+19.83	+4.61	-1.12	+1.06
Qwen2.5-7B	Vocab. Dropout	r10	+38.98	+21.49	+18.79	-1.30	+4.99
OLMo-3-7B	Vocab. Dropout	r5	+18.24	+11.76	+7.22	-0.73	+8.97
OctoThinker-8B	Vocab. Dropout	r9	+27.58	+21.35	+6.23	-0.01	+14.62
Granite-4.1-8B	Vocab. Dropout	r4	+35.97	+28.10	+9.77	-1.91	+18.34
Qwen3-4B	Human-annotated RL	s30	+28.70	+22.82	+7.26	-1.38	-0.50
Qwen3-8B	Human-annotated RL	s150	+28.85	+25.99	+4.55	-1.69	+0.27
Qwen2.5-7B	Human-annotated RL	s30	+40.97	+24.01	+18.64	-1.68	+2.64
OLMo-3-7B	Human-annotated RL	s60	+28.12	+16.96	+12.20	-1.04	+12.01
OctoThinker-8B	Human-annotated RL	s150	+30.78	+24.18	+6.31	+0.30	+15.41
Granite-4.1-8B	Human-annotated RL	s105	+39.05	+30.77	+10.53	-2.25	+25.67

$G^M = G^J + F + D$. F is the reduction in judge-correct but MathRuler-unparsed response percentage; D is the change in other correctness disagreements. $A_0 - A_t$ is the reduction in semantic abstention share. All terms average over the full evaluation set; marginal differences do not track individual response transitions.

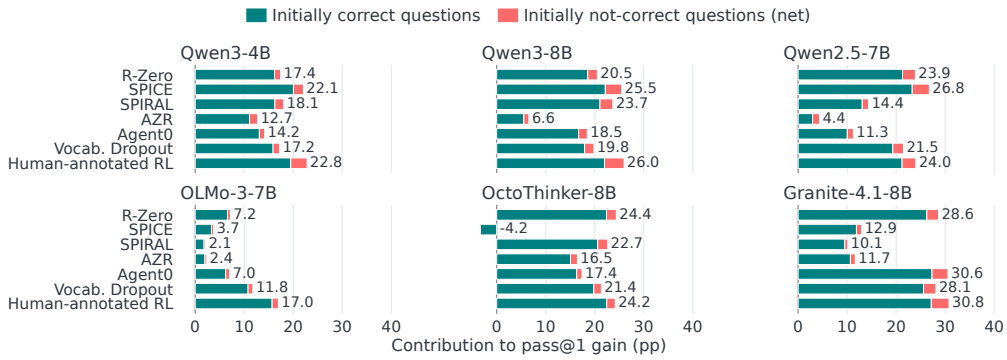

Figure 9: Peak pass@1 gain by question group, scored by the LLM judge (same checkpoints, responses, and groups as Figure 3).

B.2 Peak, vote-reference, and endpoint diagnostics

Table 5: MathRuler scores at initialization, the selected peak, and the last registered checkpoint.

Model	Recipe	Peak	pass@1 (%): init	peak	last	pass@128 (%): init	peak	last
Qwen3-4B	R-Zero	r2	37.72	62.44	54.68	88.42	87.33	72.09
Qwen3-8B	R-Zero	r2	42.21	66.06	58.71	89.32	87.86	73.80
Qwen2.5-7B	R-Zero	r2	20.12	61.27	49.81	85.52	86.71	65.77
OLMo-3-7B	R-Zero	r6	35.96	55.74	44.07	84.33	85.66	73.97
OctoThinker-8B	R-Zero	r3	22.03	52.30	46.23	82.41	80.15	66.50
Granite-4.1-8B	R-Zero	r1	37.48	73.55	69.89	88.35	87.96	82.38
Qwen3-4B	SPICE	s100	37.72	66.53	0.10	88.42	84.86	2.23
Qwen3-8B	SPICE	s100	42.21	70.16	67.36	89.32	86.74	83.43
Qwen2.5-7B	SPICE	s200	20.12	63.90	0.11	85.52	85.66	0.80
OLMo-3-7B	SPICE	s100	35.96	48.97	39.12	84.33	85.66	85.45
OctoThinker-8B	SPICE	s200	22.03	23.75	1.99	82.41	79.52	18.88
Granite-4.1-8B	SPICE	s200	37.48	56.76	9.17	88.35	90.13	82.69
Qwen3-4B	SPIRAL	s400	37.72	62.63	62.63	88.42	87.82	87.82
Qwen3-8B	SPIRAL	s400	42.21	68.71	68.71	89.32	88.00	88.00
Qwen2.5-7B	SPIRAL	s40	20.12	51.83	27.60	85.52	87.89	86.92
OLMo-3-7B	SPIRAL	s360	35.96	47.34	46.30	84.33	87.09	86.99
OctoThinker-8B	SPIRAL	s200	22.03	50.32	49.34	82.41	85.69	84.68
Granite-4.1-8B	SPIRAL	s320	37.48	49.21	48.14	88.35	89.04	88.63
Qwen3-4B	AZR	s390	37.72	54.94	54.94	88.42	88.35	88.35
Qwen3-8B	AZR	s390	42.21	50.66	50.66	89.32	89.85	89.85
Qwen2.5-7B	AZR	s260	20.12	33.97	32.53	85.52	87.47	86.95
OLMo-3-7B	AZR	s390	35.96	39.26	39.26	84.33	82.76	82.76
OctoThinker-8B	AZR	s260	22.03	42.16	30.83	82.41	80.64	74.84
Granite-4.1-8B	AZR	s390	37.48	51.73	51.73	88.35	88.59	88.59
Qwen3-4B	Agent0	s100	37.72	58.11	58.11	88.42	87.93	87.93
Qwen3-8B	Agent0	s90	42.21	63.38	63.24	89.32	88.28	88.76
Qwen2.5-7B	Agent0	s60	20.12	43.58	36.99	85.52	87.37	86.25
OLMo-3-7B	Agent0	s100	35.96	46.93	46.93	84.33	87.65	87.65
OctoThinker-8B	Agent0	s70	22.03	44.65	0.00	82.41	81.79	0.00
Granite-4.1-8B	Agent0	s80	37.48	75.06	75.01	88.35	88.70	88.66
Qwen3-4B	Vocab. Dropout	r8	37.72	62.27	61.96	88.42	88.31	88.21
Qwen3-8B	Vocab. Dropout	r10	42.21	65.53	65.53	89.32	89.01	89.01
Qwen2.5-7B	Vocab. Dropout	r10	20.12	59.09	59.09	85.52	88.35	88.35
OLMo-3-7B	Vocab. Dropout	r5	35.96	54.21	54.21	84.33	86.50	86.50
OctoThinker-8B	Vocab. Dropout	r9	22.03	49.61	48.16	82.41	85.52	84.68
Granite-4.1-8B	Vocab. Dropout	r4	37.48	73.45	69.78	88.35	88.45	90.79
Qwen3-4B	Human-annotated RL	s30	37.72	66.42	64.61	88.42	88.63	87.26
Qwen3-8B	Human-annotated RL	s150	42.21	71.06	71.06	89.32	89.39	89.39
Qwen2.5-7B	Human-annotated RL	s30	20.12	61.09	56.49	85.52	86.92	85.97
OLMo-3-7B	Human-annotated RL	s60	35.96	64.08	59.78	84.33	87.16	88.00
OctoThinker-8B	Human-annotated RL	s150	22.03	52.82	52.82	82.41	82.41	82.41
Granite-4.1-8B	Human-annotated RL	s105	37.48	76.53	76.11	88.35	90.30	90.16

All registered checkpoints have complete MathRuler scores. Peaks maximize MathRuler pass@1, with earliest-coordinate tie breaking. LLM judging uses the same selected checkpoints.

Table 6: Pass@128 at MathRuler-selected pass@1 peaks. LLM judging uses the same responses and checkpoints. Scores are percentages; ΔM and ΔJ are MathRuler and LLM-judge changes from initialization in pp.

Model	Method	Peak	Judge init.	Judge peak	ΔM	ΔJ
Qwen3-4B	R-Zero	r2	92.78	91.14	-1.08	-1.64
Qwen3-4B	SPICE	s100	92.78	89.64	-3.56	-3.14
Qwen3-4B	SPIRAL	s400	92.78	92.08	-0.59	-0.70
Qwen3-4B	AZR	s390	92.78	93.37	-0.07	+0.59
Qwen3-4B	Agent0	s100	92.78	91.91	-0.49	-0.87
Qwen3-4B	Vocab. Dropout	r8	92.78	92.04	-0.10	-0.73
Qwen3-4B	Human-annotated RL	s30	92.78	94.03	+0.21	+1.26
Qwen3-8B	R-Zero	r2	93.48	91.94	-1.47	-1.54
Qwen3-8B	SPICE	s100	93.48	91.91	-2.58	-1.57
Qwen3-8B	SPIRAL	s400	93.48	92.81	-1.33	-0.66
Qwen3-8B	AZR	s390	93.48	94.14	+0.52	+0.66
Qwen3-8B	Agent0	s90	93.48	92.78	-1.05	-0.70
Qwen3-8B	Vocab. Dropout	r10	93.48	92.78	-0.31	-0.70
Qwen3-8B	Human-annotated RL	s150	93.48	94.10	+0.07	+0.63
Qwen2.5-7B	R-Zero	r2	90.02	89.85	+1.19	-0.17
Qwen2.5-7B	SPICE	s200	90.02	89.53	+0.14	-0.49
Qwen2.5-7B	SPIRAL	s40	90.02	91.24	+2.37	+1.22
Qwen2.5-7B	AZR	s260	90.02	91.38	+1.95	+1.36
Qwen2.5-7B	Agent0	s60	90.02	91.21	+1.85	+1.19
Qwen2.5-7B	Vocab. Dropout	r10	90.02	91.59	+2.83	+1.57
Qwen2.5-7B	Human-annotated RL	s30	90.02	91.38	+1.40	+1.36
OLMo-3-7B	R-Zero	r6	91.70	89.18	+1.33	-2.51
OLMo-3-7B	SPICE	s100	91.70	92.18	+1.33	+0.49
OLMo-3-7B	SPIRAL	s360	91.70	92.60	+2.76	+0.91
OLMo-3-7B	AZR	s390	91.70	91.10	-1.57	-0.59
OLMo-3-7B	Agent0	s100	91.70	92.53	+3.31	+0.84
OLMo-3-7B	Vocab. Dropout	r5	91.70	92.57	+2.16	+0.87
OLMo-3-7B	Human-annotated RL	s60	91.70	92.11	+2.83	+0.42
OctoThinker-8B	R-Zero	r3	87.68	83.36	-2.27	-4.33
OctoThinker-8B	SPICE	s200	87.68	82.90	-2.90	-4.78
OctoThinker-8B	SPIRAL	s200	87.68	89.15	+3.28	+1.47
OctoThinker-8B	AZR	s260	87.68	85.21	-1.78	-2.48
OctoThinker-8B	Agent0	s70	87.68	86.50	-0.63	-1.19
OctoThinker-8B	Vocab. Dropout	r9	87.68	88.49	+3.11	+0.80
OctoThinker-8B	Human-annotated RL	s150	87.68	85.45	+0.00	-2.23
Granite-4.1-8B	R-Zero	r1	94.21	93.86	-0.38	-0.35
Granite-4.1-8B	SPICE	s200	94.21	95.01	+1.78	+0.80
Granite-4.1-8B	SPIRAL	s320	94.21	94.77	+0.70	+0.56
Granite-4.1-8B	AZR	s390	94.21	94.42	+0.24	+0.21
Granite-4.1-8B	Agent0	s80	94.21	95.01	+0.35	+0.80
Granite-4.1-8B	Vocab. Dropout	r4	94.21	94.73	+0.10	+0.52
Granite-4.1-8B	Human-annotated RL	s105	94.21	95.78	+1.95	+1.57

At the selected peaks, pass@128 increases in 24 settings, decreases in 17, and is unchanged in one. Each self-evolving method has both positive and negative changes across models; human-annotated RL increases on five and is unchanged on OctoThinker. Table 6 compares these changes with semantic judging at the same checkpoints. The semantic comparison is a change of scoring instrument, rather than a correction obtained by subtracting the pass@1 extraction component from pass@128. Human-annotated RL has higher peak pass@1 than every self-evolving comparator on four of six models. These are point estimates at the selected checkpoints.

Correct probability within fixed initial cohorts. Table 7 reports changes in the percentage of correct responses for initial cohorts held fixed, including initially wrong-leading questions. All methods on a model share the same initialization responses, so their initial cohort memberships

coincide. The last column reports the percentage of initially correct-leading questions that lose more than 10 pp of correct responses.

Table 7: Correct-answer changes on fixed initial cohorts at MathRuler-selected peaks.

Model	Recipe	Peak	N_c	N_w	ΔP_c	ΔP_w	Q_c down > 10
Qwen3-4B	R-Zero	r2	2075	705	+33.20	+2.42	0.2%
Qwen3-8B	R-Zero	r2	2119	678	+31.16	+3.17	0.3%
Qwen2.5-7B	R-Zero	r2	1972	775	+57.23	+5.97	0.1%
OLMo-3-7B	R-Zero	r6	2105	528	+25.53	+4.53	7.6%
OctoThinker-8B	R-Zero	r3	1771	888	+46.44	+4.16	0.5%
Granite-4.1-8B	R-Zero	r1	2270	511	+44.64	+3.43	0.0%
Qwen3-4B	SPICE	s100	2075	705	+38.54	+3.17	0.7%
Qwen3-8B	SPICE	s100	2119	678	+35.88	+5.22	1.2%
Qwen2.5-7B	SPICE	s200	1972	775	+60.15	+8.00	0.6%
OLMo-3-7B	SPICE	s100	2105	528	+17.36	+1.12	0.5%
OctoThinker-8B	SPICE	s200	1771	888	+2.23	+0.85	22.6%
Granite-4.1-8B	SPICE	s200	2270	511	+23.31	+3.88	6.6%
Qwen3-4B	SPIRAL	s400	2075	705	+33.08	+3.40	0.3%
Qwen3-8B	SPIRAL	s400	2119	678	+34.49	+3.74	0.5%
Qwen2.5-7B	SPIRAL	s40	1972	775	+44.51	+3.66	0.0%
OLMo-3-7B	SPIRAL	s360	2105	528	+14.88	+1.97	0.8%
OctoThinker-8B	SPIRAL	s200	1771	888	+42.94	+4.85	0.0%
Granite-4.1-8B	SPIRAL	s320	2270	511	+14.52	+1.08	0.3%
Qwen3-4B	AZR	s390	2075	705	+22.67	+2.95	0.0%
Qwen3-8B	AZR	s390	2119	678	+10.68	+2.11	0.6%
Qwen2.5-7B	AZR	s260	1972	775	+18.75	+3.16	0.0%
OLMo-3-7B	AZR	s390	2105	528	+4.39	+0.30	11.8%
OctoThinker-8B	AZR	s260	1771	888	+30.81	+2.91	0.7%
Granite-4.1-8B	AZR	s390	2270	511	+17.57	+1.51	0.7%
Qwen3-4B	Agent0	s100	2075	705	+27.41	+1.94	0.0%
Qwen3-8B	Agent0	s90	2119	678	+27.73	+2.46	0.0%
Qwen2.5-7B	Agent0	s60	1972	775	+32.80	+3.05	0.0%
OLMo-3-7B	Agent0	s100	2105	528	+13.92	+3.23	3.0%
OctoThinker-8B	Agent0	s70	1771	888	+35.21	+2.33	0.5%
Granite-4.1-8B	Agent0	s80	2270	511	+46.14	+4.36	0.1%
Qwen3-4B	Vocab. Dropout	r8	2075	705	+32.82	+2.83	0.2%
Qwen3-8B	Vocab. Dropout	r10	2119	678	+30.41	+3.16	0.2%
Qwen2.5-7B	Vocab. Dropout	r10	1972	775	+54.22	+5.51	0.1%
OLMo-3-7B	Vocab. Dropout	r5	2105	528	+24.18	+2.02	0.0%
OctoThinker-8B	Vocab. Dropout	r9	1771	888	+42.14	+4.21	0.0%
Granite-4.1-8B	Vocab. Dropout	r4	2270	511	+44.15	+4.57	0.0%
Qwen3-4B	Human-annotated RL	s30	2075	705	+36.79	+7.68	0.6%
Qwen3-8B	Human-annotated RL	s150	2119	678	+35.89	+9.00	0.7%
Qwen2.5-7B	Human-annotated RL	s30	1972	775	+56.84	+6.31	0.1%
OLMo-3-7B	Human-annotated RL	s60	2105	528	+36.52	+5.09	0.0%
OctoThinker-8B	Human-annotated RL	s150	1771	888	+46.64	+5.48	2.1%
Granite-4.1-8B	Human-annotated RL	s105	2270	511	+46.72	+9.02	0.5%

Cohorts use unique initial leaders. Probability changes are in pp; the last column is the fraction of initially correct-leading questions losing more than ten points.

Final-checkpoint diagnostics. Table 5 reports available scores at the last registered checkpoint separately from each confirmed peak.

B.3 Checking the reasoning of correct responses on AIME

Pass@128 counts a question as solved if any of its 128 final answers is correct, which a lucky answer can satisfy on hard questions. Following Wen et al. (2026), we also check whether the reasoning of correct responses establishes the answer, and count a question as solved only if at least one response does (CoT-Pass@128). We run this check on the 60 AIME 2024 and 2025 questions for Qwen3-8B at initialization and at the peaks of R-Zero, SPICE, SPIRAL, AZR, Agent0, Vocabulary Dropout, and human-annotated RL, using the saved responses whose final answer MathRuler marks correct. For each question and checkpoint we first check up to three such responses chosen at random; where none is valid, we check up to five more and then all remaining ones. Every count in Table 8 is therefore exact. Claude Opus 5.5 judges each response without knowing its checkpoint; it judged 1,083 responses in total.

The main count treats correct code whose output the model only claims, without being able to run it, as invalid; the lenient count accepts these 19 responses. Invalid responses usually reach the right answer through errors that cancel. On one AIME 2025 problem, many responses intersect the rotated parabola with the wrong equation, whose roots are the negatives of the true ones and give the same

Table 8: Questions solved on AIME 2024 and 2025 (60 questions), Qwen3-8B.

Checkpoint	pass@128	CoT-Pass@128	CoT-Pass@128 (lenient)
Initialization	38	16	18
R-Zero	30	16	16
SPIRAL	33	19	20
AZR	36	22	24
Agent0	31	17	18
Vocabulary Dropout	32	16	19
SPICE	26	18	18
Human-annotated RL	37	25	25

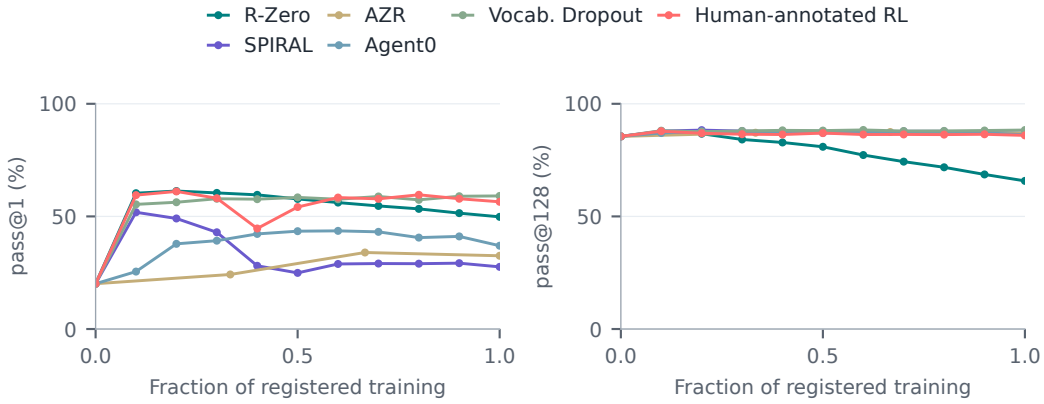

Figure 10: Accuracy and coverage on Qwen2.5-7B.

final answer. Other common cases are area formulas whose errors cancel, symmetry assumed without proof, and guessed patterns.

Judge instructions. The judge sees the problem, the reference answer, and the response, and is told that the final answer is correct. A response is *valid* if it derives the answer through correct reasoning in which every key step is justified; minor slips that are corrected or do not affect the result are acceptable, and checking a guessed answer counts only if the check itself proves the answer, as an exhaustive verification does. It is *invalid* if the answer rests on a critical error, a false or unjustified claim, pattern-guessing, an unverified assumption, or a computation that actually gives a different value, or if the response states the answer without a derivation or stops before establishing it. The judge solves or checks the problem itself as needed.

C Training Trajectories

C.1 Cross-model trajectories

Models and evaluation schedules. We select six dense base-model initializations with 4–8B parameters to cover multiple model families and levels of initial mathematical proficiency. This range permits repeated training and evaluation across methods. The models are Qwen3-4B, Qwen3-8B, Qwen2.5-7B, OLMo-3-7B, OctoThinker-8B, and Granite-4.1-8B; OctoThinker uses its mid-trained Hybrid-Base checkpoint. The checkpoint grid contains ten trained checkpoints per model for R-Zero, SPIRAL, Agent0, and human-annotated RL; three for AZR (steps 130, 260, and 390); and ten for Vocabulary Dropout except OLMo, which has five. SPICE adds its checkpoints every 100 steps on all six models (up to step 200 on OLMo-3-7B) (Appendix E.2). This yields 350 registered trained checkpoints, excluding the six shared initializations. Checkpoint coordinates follow each method's native schedule and do not imply equal training budgets. Because of compute constraints, Vocabulary Dropout on OLMo-3-7B runs five rounds, the default in the method's released code, rather than ten, so its curves in Figure 2 end at round 5. Its pass@1 still rises at round 5, so its peak (round 5) may understate what ten rounds would reach.

Coverage moves in both directions at the accuracy peak: it rises in 24 of the 42 confirmed settings and falls in the 17; one is unchanged. Table 5 reports initialization, peak, and endpoint scores for this frozen comparison. Figure 2 separately reports available training trajectories under rule-based scoring on the full 2,866-question set.



Figure 11: Full Agent0 training trajectories.

Similar peaks, different retention. The complete Qwen2.5 trajectories show early accuracy improvements followed by different retention patterns. R-Zero loses part of its early gains, while Vocabulary Dropout retains higher late-training accuracy (Figure 10). These curves report the full evaluation with the rule-based grader; primary peaks are selected from these complete registered evaluation grids.

Figure 10 normalizes training coordinates by each method's registered horizon; Figure 2 uses native coordinates and common zero-based percentage scales. Figure 11 retains all Agent0 checkpoints, including the sharp decline on OctoThinker after step 70. Figure 2 displays the steps 0–100 for all six Agent0 models, with downward triangles marking OctoThinker values below the displayed vertical ranges and annotations giving its step-100 changes. The appendix retains the full vertical ranges. The main figure shows percentage-point changes from initialization, whereas the appendix figure shows absolute scores. The methods behave differently after the peak. R-Zero ends below its peak on every model. The other methods, including human-annotated RL, keep improving until the last checkpoint in some settings and decline in others: AZR, for example, still improves at step 390 on Qwen3-4B and Qwen3-8B but declines after step 260 on Qwen2.5-7B and OctoThinker-8B. Endpoint and peak scores from the rule-based grader appear in Table 5.

D Loop interventions and supervision

D.1 Changing the question source inside the R-Zero loop

All four conditions in Section 4.1 run ten R-Zero rounds on Qwen3-8B with the same solver recipe: ten votes per question, majority-vote labels from the current solver, the $[0.3, 0.8]$ agreement band, and 15 solver updates per round. Labels and the agreement band are recomputed every round, including for the reused and perturbed question sets. They differ only in the source of questions. The full loop has three runs for rounds 1–3, two for rounds 4–6, and one for rounds 7–10; the frozen proposer has three runs; the reused and perturbed question sets have one run each. The full loop trains the proposer each round. The frozen proposer is the untrained base model and writes a new question set each round. The reused question set is the set generated in round one, used again unchanged. The perturbed question set is the same round-one set with every standalone integer in the question text shifted by an offset drawn from $\{-3, -2, -1, +1, +2, +3, +5, +7\}$, with a fixed seed. Questions contain 3.8 integers on average; 11.1% contain none and are unchanged. Every checkpoint is evaluated with the protocol of Section 3.1.

Question-validity audit. We draw 500 questions at random from each round-one question set, shuffle the 1,000 questions together, and hide their source. Claude Opus 5.5 labels each question as valid, invalid, or unsure, and solves it when it is valid. We run two independent passes with different batch orders. The passes agree on 97.2% of original and 95.8% of perturbed questions. Both passes mark 151 original questions (30.2%) and 280 perturbed questions (56.0%) as invalid; the difference is 25.8 pp (95% CI 19.9–31.7). Where the auditor could solve the question, the training label matches its answer for 197 of 299 original questions and 109 of 168 perturbed questions.

D.2 Round-by-round task audit

Math questions. The audit covers three Qwen3-8B runs: R-Zero, R-Zero with the frozen proposer of Section 4.1, and Vocabulary Dropout. For each run and round we draw 300 questions at random, without overlap, from the question set retained for solver training, and audit them in three blind

passes of the same design. Within a pass, questions are shuffled together and shown without run, round, or label. Claude Opus 5.5 marks each question as valid, invalid, or unsure, and gives a final answer for valid questions it can solve. A question is invalid if its conditions contradict each other, are missing, or do not determine a unique answer. We compare the answer with the training label after normalizing numbers and expressions; every remaining mismatch is checked again for equivalence, for example $45/14$ and 3.214. Table 9 lists the counts.

Table 9: Round-by-round audit of retained math questions (300 per cell). Each cell gives invalid questions, then correct labels out of solved valid questions.

Round	1	2	3	4	5	6	7	8	9	10
R-Zero, invalid	109	122	158	174	143	147	167	179	182	184
R-Zero, labels	139/182	86/168	55/130	47/117	48/145	43/141	31/126	29/112	22/114	26/111
Frozen proposer, invalid	103	122	139	131	131	123	132	146	161	140
Frozen proposer, labels	138/195	102/170	77/154	77/162	62/164	67/173	58/162	42/152	40/137	55/156
Vocabulary Dropout, invalid	118	238	217	250	242	241	244	246	240	253
Vocabulary Dropout, labels	136/182	23/62	36/83	21/50	22/58	29/59	18/56	21/54	13/60	16/47

Final answers equal to 0 or 1 on other models. Without any model review, we count the percentage of retained questions whose final answer (label) is 0 or 1 in round 1 and round 10. For R-Zero it rises on all five models we could check (OctoThinker-8B's R-Zero question sets were not available for this count): Qwen3-8B 16%→39%, Qwen3-4B 21%→68%, Qwen2.5-7B 19%→34%, OLMo-3-7B 38%→70%, and Granite-4.1-8B 5%→28%. With the frozen proposer on Qwen3-8B it rises from 19% to 32%. For Vocabulary Dropout it stays between 3% and 5% on all six models.

AZR code tasks. We split the first 280 of the 390 AZR training steps on Qwen3-8B into ten consecutive 28-step windows and draw 150 output-prediction tasks from the buffer of each window. The reviewer runs each program on its input, compares the result with the stored output, and marks the task as trivial when the input skips the computation the program describes, for example an empty list or an input that never enters the main branch. Trivial tasks per window are 27, 18, 7, 10, 4, 8, 1, 3, 1 and 0 out of 150. Every stored output matches execution, except for three programs in the first window that cannot run or give no fixed output. The percentage of distinct inputs among all output tasks falls from 79% in the first window to 37% in the third, where the input `[1, 2, 3, 4, 5]` alone makes up 26% of tasks, and recovers to 56% by the last window.

SPIRAL games. SPIRAL saves the games played during training, so we also examine its self-play games (all Kuhn Poker, 51,264 games and about 218,000 hands across eight runs). In the environment version we used, a bet or a call does not change the pot, so only folding costs chips. Under these payoffs, betting and calling on every hand is optimal. All runs move to this policy within the first 10–20% of training: in the last 10% of training, 98–100% of hands are bet and called, folds fall from 2–28% to under 2%, and responses become shorter. Under standard Kuhn Poker payoffs, the resulting policy is exploitable by about 0.33 chips per hand. The self-play opponents that the fixed-solver comparison uses from later in training therefore play nearly the same strategy.

Judge prompts. The judge received the following instructions for each batch of 100 items, whose source was hidden; file paths and output-handling instructions are omitted.

Math questions.

> You are a careful mathematician auditing generated math problems for a research paper. For every problem decide, reading it exactly as written (do not fix typos or guess an intended version): "valid": conditions consistent and satisfiable, question well defined, single determinate answer. "invalid": conditions contradict or cannot all hold; a required quantity is undefined or the question is ill-formed; the answer is not unique or not determined; or it asks for something that does not exist. An answer of 0, "no such object" or "infinitely many" is not by itself invalid if it is the determinate answer to a well-posed question. "unsure": cannot decide with reasonable effort. For valid problems give your final answer if you can determine it with reasonable confidence, else null. Check numbers and conditions explicitly. Return for each problem: class, a one-sentence reason, and the answer.

Code tasks (AZR).

> Each task asks a solver to predict the output of the Python function `f` when called on the given input. For every task decide its quality: "meaningful": `f` performs a non-trivial computation that the given input actually exercises, so predicting the output requires reasoning through the code. "trivial": the output can be read off without real computation (e.g., `f` returns its input or a constant, the input bypasses the main logic, or the relevant branch is never reached). "broken": the program cannot run on this input, is non-deterministic (randomness, time, hashing order), or is otherwise ill-posed. Also decide whether the stated output equals what

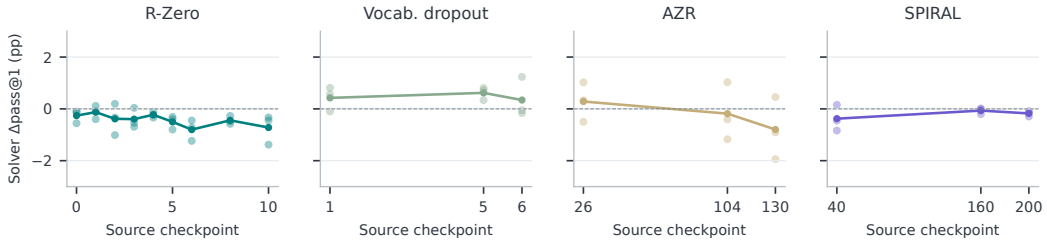

Figure 12: Fixed-student learning utility across all available source checkpoints for four methods. Small dots show all three runs; curves show their mean change from the method's starting solver. R-Zero and Vocabulary Dropout use proposer rounds; AZR uses task-source steps and SPIRAL uses opponent steps. All four methods use Qwen3-8B. Student-update budgets are matched within each method. AZR and SPIRAL outcomes measure mathematical transfer. The 54 individual runs are listed in Table 11.

> `f` returns (true, false, or null). You may execute the code only in a sandboxed subprocess with a 10-second timeout and no file, network, or system access.

Label equivalence.

> You are checking whether two final answers to a math problem are mathematically the same answer. Treat notation differences as the same (e.g., a binomial expression vs its evaluated integer, "$x = 5$" vs "5", 0.5 vs 1/2). Compute numerically when needed. Different values, or one side being a non-answer (e.g., "proof"), count as different. You do not judge which one is correct.

D.3 Proposer comparison with a fixed starting solver

We compare checkpoint age while fixing the student within each method. The reported experiments comprise 27 R-Zero continuations, nine each for Vocabulary Dropout, AZR, and SPIRAL, and six for SPICE. Agent0 is not included: in its Qwen3-8B run the proposer is trained once, for six steps, before 100 solver steps, so there is no early and later proposer to compare. All outcomes below use the same 2,866-question mathematical evaluation, 128 samples per question, and a 4,096-token output cap. Pass@1 is the mean correct fraction across questions. Table 11 reports all runs, including the intermediate snapshots and the extended R-Zero sweep. The three replications hold the source checkpoint fixed; they do not replicate training the original source checkpoint.

SPICE: fixed student and fresh tasks. The SPICE comparison resumes the Qwen3-8B solver of the SPICE run at step 300, with its optimizer state, and trains it for 100 more solver steps on tasks from the same run's proposer at step 100 (early) or step 700 (later). Each proposer writes its tasks from documents with SPICE's own generation procedure; for a given seed, both proposers draw the same candidate documents, and each solver step uses one task per document with eight answers per task, as in the original run. Seeds vary the document order, task generation, label sampling, and data order. Only the solver is updated, as in the AZR runs. Tasks from the later proposer lower pass@1 relative to the early proposer in all three runs, by 2.5, 1.4, and 18.1 pp. The later proposer's tasks list four answer options without a question, and the correct option is always A; the students learn to answer "A" and give a bare "A" on 2.2%, 0.2%, and 15.3% of the evaluation samples, against at most 0.01% for the starting solver and the early students. The SPICE run itself shows the same pattern: from step 50 on, 99% of its multiple-choice tasks have no question, and from step 401 the correct option is always A.

R-Zero: fixed student and fresh question sets. The proposer comparison starts every branch from the Qwen3-8B solver at the end of round five in the R-Zero run. We compare the proposer after round one and the proposer updated for round six. The latter is trained against the round-five solver before generating experience for the next solver phase. Both proposers generate fresh questions for this experiment. The round-one proposer has already been trained. This is a cumulative early-versus-updated proposer comparison; a round-five versus round-six proposer contrast measures the additional effect of that update for the same student. The extended sweep evaluates proposer rounds $0, 1, 2, 3, 4, 5, 6, 8, 10$, where round zero is the untrained proposer.

Each branch makes 8,000 generation attempts. The fixed student supplies ten answer votes per question; nonempty questions and labels with agreement in $[0.3, 0.8]$ are retained under the same procedure. Three paired runs vary the question-generation, label-sampling, and data-order seeds; the training-rollout seed remains fixed. The accepted question sets have different sizes, reflecting the

Table 10: Solver performance after learning from each proposer, by paired run.

Proposer	Run	Retained tasks	pass@1 (%)	pass@128 (%)	Plurality (%)
Starting solver θ_5	–	–	64.25	83.39	69.30
Early proposer ϕ_1	1	3,628	64.37	81.02	68.32
Early proposer ϕ_1	2	3,659	64.15	81.33	68.11
Early proposer ϕ_1	3	3,709	63.85	81.51	67.97
Updated proposer ϕ_6	1	4,130	63.01	80.60	67.06
Updated proposer ϕ_6	2	4,268	63.53	80.60	66.85
Updated proposer ϕ_6	3	4,230	63.80	81.02	67.76

native generation and selection process. The early proposer yields 3,628, 3,659, and 3,709 questions; the updated proposer yields 4,130, 4,268, and 4,230. Generation attempts and solver-update budgets are matched, rather than the sizes of the retained sets.

All six branches start from identical student weights with a fresh optimizer and the same frozen KL reference. Each completes 15 updates with 512 prompt groups and five responses per group: 7,680 prompt exposures and 38,400 training responses. The AdamW learning rate is 10^{-6} and the KL coefficient is 0.01. The training inputs are generated questions and student-voted labels; evaluation questions and answers are not supplied to the generation or training pipeline.

We evaluate the starting student and all six trained checkpoints on the complete 2,866-question set with 128 samples per question and the 4,096-token output budget, using the same rule-based grader as Figure 2. Table 10 reports every run. Accuracy utility averages the three per-run changes from the common starting student. Pass@128 is computed from 128 samples separately for each run and then averaged across runs. Figure 6b shows mean learning utility for the early, intermediate, and later sources; panel c shows each paired later-minus-early difference and its mean. These are observed run differences, not confidence intervals over training replications.

Vocabulary Dropout: proposer-stated labels. We fix the round-five Qwen3-8B solver and compare proposers from rounds one, five, and six, generating a fresh question set for each run. Each condition uses 32,000 generation attempts, ten votes, the $[0.3, 0.7]$ agreement band, and 20 solver updates. Generation-time vocabulary dropout remains active, and proposer-stated answers are retained as training labels. Seed settings are paired across source checkpoints; accepted set sizes are outcomes of the shared generation and selection procedure, not matched inputs.

AZR: frozen task sets and solver-only updates. The fixed Qwen3-8B student is the step-130 checkpoint. Task sources are the checkpoints at steps 26, 104, and 130, each with its own task buffer and native execution checks. A separate offline task set is generated once per source; its three solver replications consume the same tasks. Each continuation performs 26 solver-only updates with 64 tasks of each of the three task types per update, totaling 4,992 task exposures without replacement. Generation continues until every type has at least 1,664 accepted tasks. This takes 62, 29, and 28 generation rounds for sources 26, 104, and 130, respectively, with 192 proposals per round. Thus accepted training-task budgets are matched, but proposal budgets differ. These comparisons change the frozen task source, including its buffer; they do not isolate proposer weights from buffer history. Because the native algorithm shares parameters across roles, solver-only training on offline task sets also differs from native joint updating.

SPIRAL: frozen opponents. The Qwen3-8B learner starts at step 200 and trains for 40 updates against a frozen opponent from steps 40, 160, or 200. All conditions use Kuhn poker with five rounds and 128 learner-turn samples per update. Seats are balanced and game seeds paired; only learner turns enter the update. The opponent's weights remain fixed. These continuations test the mathematical transfer of learning against different opponents, not the effect of continuously updating both seats. AZR and SPIRAL continuations start with fresh optimizer states in every condition.

Question-level differences by starting difficulty. Figure 13 expands the later-minus-early comparison to individual evaluation questions. For each question, we average the difference in correct-response fraction over the three paired runs and express it in pp. We group questions by the starting solver's observed correct count among 128 responses: zero, 1–32, 33–96, and 97–128. Every method retains all 2,866 questions; cohort membership is fixed by its own starting solver. The mean over these question effects reproduces the method's reported later-minus-early difference. The distributions describe variation across questions, not independent proposer-training replications or an explanation of why effects differ. An observed zero count does not establish zero true success probability.

What the comparisons establish. R-Zero's mean utility is negative at every sampled proposer checkpoint, with non-monotonic variation across rounds. Vocabulary Dropout has positive mean

Table 11: Fixed-student runs: post-training pass@1 by run, mean change from the starting solver (U, pp), and mean pass@128 (%). Sources are proposer rounds for R-Zero and Vocabulary Dropout, and training steps for AZR and SPIRAL. Starting pass@1 is 64.25, 64.42, 47.61, and 68.51%, respectively.

Method	Source	Run 1	Run 2	Run 3	Mean U	Mean pass@128
R-Zero	0	64.13	64.14	63.69	−0.26	81.32
	1	64.37	64.15	63.85	−0.13	81.29
	2	64.44	63.24	63.91	−0.38	80.72
	3	64.29	63.56	63.71	−0.40	80.88
	4	64.01	63.91	64.14	−0.23	80.89
	5	63.45	63.94	63.85	−0.50	80.74
	6	63.01	63.53	63.80	−0.80	80.74
	8	63.67	63.98	63.77	−0.44	80.70
	10	62.87	63.80	63.92	−0.72	80.48
Vocabulary Dropout	1	65.24	64.32	64.98	+0.42	89.20
	5	65.14	64.76	65.23	+0.62	89.33
	6	64.37	64.26	65.66	+0.34	89.13
AZR	26	47.11	48.64	47.94	+0.28	89.61
	104	46.44	47.21	48.64	−0.18	89.56
	130	45.67	46.71	48.07	−0.80	89.81
SPIRAL	40	67.67	68.06	68.66	−0.38	88.32
	160	68.49	68.31	68.52	−0.07	87.97
	200	68.36	68.42	68.21	−0.18	88.30

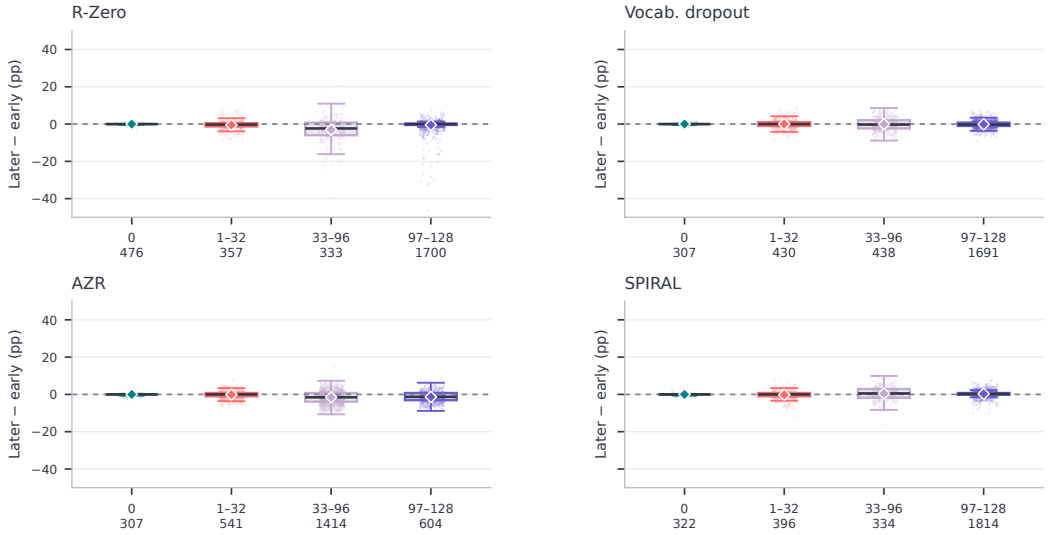

Figure 13: Which questions benefit from replacing the early proposer or opponent with the later one? Positive values favor the later source. Each dot is one question's mean difference over three paired runs; boxes show the middle 50%, horizontal lines the median, whiskers extend to observations within 1.5 times the interquartile range, and diamonds show means. Initial correct counts define the four bins; question counts appear underneath. All four methods use Qwen3-8B with rule-based scores from 128 responses per question and run and a 4,096-token cap.

utility at all three snapshots without a consistent later-versus-early advantage. AZR's later source reduces mathematical pass@1 relative to its early source in two of three runs; SPIRAL's paired effects have mixed signs. The cross-method comparison therefore tests whether source age predicts downstream learning, without treating different recipes or native training domains as a matched causal intervention.

D.4 Errors in generated training tasks

The expanded review covers 700 records: 200 each for R-Zero and Vocabulary Dropout, and 100 each for Agent0, AZR, and SPIRAL. The records span 27 available method–model combinations

across six models. An initial 200-record Qwen3-8B review was followed by 500 records sampled across methods and models. These strata have different sizes and provenance; we use the review to identify error types, rather than estimate a pooled error rate. Mathematical checks and static program inspection support the examples below.

Distinct errors require distinct checks. We distinguish incompatible or missing task conditions, incorrect answers, erroneous supplied solutions, and mismatches between task descriptions and the computations exercised by stored tests. These categories can overlap. A valid problem can carry an incorrect label or solution; a label consistent with program execution does not establish that the tests exercise the task's stated computation. An answer of zero, or infinitely many solutions, is not itself a defect.

Table 12: Verifiable defects in retained records. Examples establish occurrence, not prevalence.

Setting	Defect	Independent check
R-Zero / Qwen3-8B	Inconsistent conditions	An integer polynomial divisible by $x^2 - 4$ must have $f(1)$ divisible by three, contradicting the specified $f(1) = 2024$.
R-Zero / Qwen3-8B	Incorrect label	For a regular pentagon on a circle of radius R, the sum of squared distances from another point on the circle to its vertices is $10R^2$, not the stored $5R^2$.
Vocabulary Dropout / Qwen3-8B	Incorrect solution and label	Selling one third of 60 fruits, then two fifths of the remainder, leaves $60(2/3)(3/5) = 24$; the supplied solution and label give 36.
AZR / Qwen3-4B	Untested stated computation	A graph-diameter record contains ten inputs that never enter the graph-building branch. The graph remains empty and all stored outputs are zero. These tests can be satisfied without computing a nonempty graph's diameter.

The first three examples admit direct mathematical checks. The AZR example is based on reading the full program and inputs; we did not execute generated code. It characterizes the stored tests, not all possible inputs. The review does not trace whether each individual record was sampled for a gradient update, and these examples do not identify the cause of a measured solver loss.

The expanded audit establishes concrete reliability defects; changes in their prevalence and their effects on learning remain separate measurements.

D.5 Original R-Zero evaluation protocol

A separate greedy-decoding evaluation uses the original seven-suite macro metric and grader version. Scores at initialization and rounds 1–10 are 38.92, 43.21, 47.02, 43.21, 40.51, 41.15, 38.69, 37.86, 36.30, 35.81, and 34.40. The run peaks at round two and ends 4.52 pp below initialization. These greedy scores assess replication under that protocol; they are not pooled with retained-sample semantic pass@1. The paper/code settings are recorded in Appendix E.2.

E Evaluation protocol and reproducibility

E.1 Evaluation setting and experiment taxonomy

Fixed labels within rounds and regenerated question sets. GRPO resamples solver *rollouts* while holding each realized label fixed. Because Q_t is regenerated each round, label-source analysis concerns the population of generated $(q, \hat{a})$ pairs, rather than repeated voting on a persistent question.

Scoring coverage. The checkpoint grid contains 350 trained checkpoints across 42 settings, all scored with the rule-based grader. All 42 pass@1-selected peaks and their initializations also have complete semantic judgments for the fixed-checkpoint sensitivity comparison.

E.1.1 Self-evolving RL

Scope. We study the language-model recipes in Table 1. Self-vote methods train against the solver's own majority-vote labels, as analyzed in §F. Methods with external checks compute rewards from verifiable outcomes on generated proxy tasks; those outcomes do not directly provide labels for evaluation questions.

In our R-Zero implementation, the proposer is first updated against the current solver using its uncertainty score and repetition penalty. The updated proposer then generates 8,000 candidate questions for the solver phase. The current solver answers each $m = 10$ times; the plurality answer becomes the pseudo-label, retained when its agreement lies in $[0.3, 0.8]$, where agreement is the largest answer count divided by the number of parsed votes. The solver trains with GRPO (Shao

Algorithm 1 One round of the proposer–solver learning loop.

State: solver π_{θ_t}; proposer μ_{ϕ_t}
Rules: candidate budget B_{cand}; proposer score R_{P}; supervision rule L; selection rule K; solver reward R_{solver}
P1 — Update the proposer
$\phi_{t+1} \leftarrow \text{UpdateProposer}(\phi_t; \pi_{\theta_t}, R_{\text{P}})$
P2 — Generate candidate tasks
$q_i \sim \mu_{\phi_{t+1}} \quad (i = 1, \ldots, B_{\text{cand}})$
P3 — Construct supervision and select examples
$z_i \leftarrow L(q_i; \pi_{\theta_t}, \mu_{\phi_{t+1}})$ ▷ method-specific labels or verification information
$\mathcal{D}_t \leftarrow \{(q_i, z_i) : K(q_i, z_i) = 1\}$
P4 — Update the solver
$\theta_{t+1} \leftarrow \text{RL}_{\text{S}}(\theta_t, \mathcal{D}_t; R_{\text{solver}})$

et al., 2024) (group size $G = 5$, 15 configured optimizer steps, batch 128) against the frozen labels, with the configured 9:1 answer/format reward mixture. No human-annotated label enters this loop.

Reference configuration. R-Zero's paper and released code differ on five settings. We follow the paper on four of them and the code on one (complete audit in Appendix E.2, Table 13). We follow the *paper* for the proposer optimizer updates per iteration (20, against the code's 192), the solver's configured maximum steps per iteration (15, against the code's 20), the proposer batch and steps (128/5, against the code's 16/6), and the label vote count (m=10, against the code's default m=9). We follow the *code* for the difficulty band on the plurality fraction, $[0.3, 0.8]$, against the paper's $[0.3, 0.7]$. We report two evaluation protocols. The original greedy R-Zero evaluation (§D.5) assesses replication fidelity through direct comparison with the published table. The retained-sample n=128 evaluation (§E.1.2) supports plurality estimates, pass@k curves, and per-question answer accounting, which cannot be computed from greedy decoding alone.

Experiment taxonomy. The primary comparison evaluates the recipes in Table 1 as complete training systems under their native schedules and budgets. Human-annotated RL changes both the question source and the labels.

E.1.2 Experimental setting

The primary solver evaluation uses a fixed mathematics question set without subsampling. Each checkpoint answers the same questions with the specified sample and output budgets; scorer revisions are recorded below. Retaining every response supports comparisons across rounds and per-question analysis. The 2,866 questions comprise OlympiadBench-EN (675), MATH-500 (500), Minerva-Math (272), AMC (40), AIME24 (30), AIME25 (30), and the full GSM8K test split (1,319). The peak analysis covers all 42 settings. Each admitted checkpoint has 2,866 questions and 128 retained samples per question (366,848 outputs). The evaluation uses n=128, temperature 1.0, top-p 1.0, top-k 40, a 4,096-token output cap, and a fixed prompt requiring step-by-step reasoning and a final `\boxed{}` answer.

Full-question training trajectories. Figure 2 uses the same full 2,866-question set, with 128 samples and the 4,096-token output cap. It contains all scored trained checkpoints of the other methods, retained in the main figure and the full Agent0 appendix figure. SPICE is evaluated every 25 steps up to step 200 and every 100 steps after that; its panel shows steps 25, 50, 100, 200, 400, and 700 (up to step 200 for OLMo-3-7B), which are evenly spaced on its logarithmic step axis. Each point has all 366,848 outputs graded with the fixed boxed-answer extractor and rule-based answer grader against the benchmark references. Pass@1 is the mean correct-sample fraction over questions; pass@128 is the fraction with at least one correct sample.

The main figure reports changes from initialization and retains each recipe's native training coordinate. One shared initialization evaluation is used per model. AZR is evaluated at steps 130, 260, and 390. Solid lines connect measured checkpoints. The main figure shows all Agent0 steps through 100; downward triangles mark the late OctoThinker values below the displayed vertical ranges, with step-100 changes annotated. Appendix Figure 11 retains the full vertical ranges. The figure uses the rule-based grader throughout; the primary answer-class diagnostics use the same rule-based scores, supplemented by fixed-checkpoint semantic judgments. At Agent0's final OctoThinker checkpoints, almost all responses lack a parsable boxed answer, yielding near-zero rule-based scores.

The released initialization checkpoints are `Qwen/Qwen3-4B-Base`, `Qwen/Qwen3-8B-Base`, `Qwen/Qwen2.5-7B`, `allenai/Olmo-3-1025-7B`, `OctoThinker/OctoThinker-8B-Hybrid-Base`, and `ibm-granite/granite-4.1-8b-base`.

We report three complementary performance measures: correctness of a single response (pass@1), correctness of at least one response among repeated samples (pass@k), and correctness of the most frequent sampled answer (maj@k). The population mode is unobserved, so we estimate it using the realized 128-sample plurality and interpret small differences in light of sampling uncertainty. The resampled maj@128 estimator is reported separately where used.
Evaluation questions are not supplied to the mainline or component-training pipeline. Comparisons use a shared evaluation setting within model families.

Semantic scoring. The scorer version is recorded for every checkpoint; some initialization and peak pairs were judged by different scorer versions. A versioned scoring cascade processes every retained output. A pinned deterministic parser and mathematics grader score unambiguous answers, and a brace-aware rule recovers explicit final boxed answers missed by the original parser. Claude Haiku 4.5 processes the remaining outputs in question-level batches, extracting only a committed final answer and returning ABSTAIN otherwise. After deterministic equivalence checking, two independent batched Haiku screens examine the answer classes. Non-unanimous or conflicting cases receive three independent judgments, with disagreements escalated to Claude Opus 5. Scorer versions record stage-specific fallbacks: extraction refusals can become abstentions, undecidable equivalence can retain a non-equivalence decision, and partial wrong-answer partitions can be completed conservatively. Other unresolved failures prevent inclusion of the checkpoint summary. The scorer processes all samples without judge subsampling or extrapolation. Pass@k, Cover@τ, leading-answer transitions, and gain accounting are recomputed from the same versioned ledger. A seeded audit of deterministic positives tests false positives without selectively changing the primary score. This cascade uses model judgment only where deterministic scoring is insufficient, reducing both cost and the judge dependence cautioned by Zheng et al. (2023).

E.2 PAPER–CODE CONFIGURATION AUDIT

SPICE. SPICE (Liu et al., 2025) uses one shared policy to generate document-grounded tasks and solve them without access to the source document. No official code is released, so we implement SPICE from the paper, with one model in both the Challenger and Reasoner roles. We use the paper's prompts, Challenger and Reasoner rewards, Dr. GRPO objective without a KL term, learning rate 10^{-6}, 128 documents per step, groups of 8, and up to 1,024 attempts per document; as in the paper's prompts, multiple-choice options are not shuffled. The paper's math documents (Nemotron-CC-Math) require access approval and its general documents (from NaturalReasoning) are not released, so we draw documents from DCLM-baseline (Li et al., 2024) and FineMath-4plus (Ben Allal et al., 2025) instead. Each model trains for 700 steps (the paper uses 640), except OLMo-3-7B, which we stop at step 200 because its training is very costly: one step takes about 13 minutes on eight GPUs, so 700 steps would need about 1,200 GPU-hours, about five times as many as on Qwen3-8B. We evaluate checkpoints under the same protocol as the other methods, every 25 steps up to step 200 and every 100 steps after that, and select the peak among the seven checkpoints every 100 steps. Pass@1 peaks at step 100 or 200 and then falls (Figure 2), to below 5% on Qwen3-4B, Qwen2.5-7B, and OctoThinker, and to 9% on Granite.

SPIRAL terminal reward. In SPIRAL's Kuhn poker environment, the game instructions define victory by cumulative chips over five rounds, but the terminal reward follows the final-round winner. Replaying the saved Qwen2.5-7B records identifies 2,881 equal-stakes, five-round bet/call games; all reward the final-round winner, and 936 disagree with the cumulative-chip winner. The records store only the last game of each actor step, so these counts describe that identifiable subset. The pinned source's final-round branch bypasses its cumulative-chip comparison. Its effect on learning is not measured here.
Table 13 compares the original R-Zero paper (Huang et al., 2026), its released code, and our replication. Every entry was verified against the relevant file and line in both codebases. The released code differs from the paper on five settings; we follow the paper on four. The paper's statement that reported results come "after 45 training steps," with figure checkpoints every 15, is arithmetically consistent only with the paper's configuration (3×15; the released script would give $3 \times 20 = 60$). Our configuration differs from the paper only in the difficulty band's upper edge (0.8 versus $[3, 7]/10$ in the text), where it follows the released code.

F KL-PROXIMAL ANALYSIS OF LABEL-DIRECTED REINFORCEMENT

We analyze how label selection determines reinforcement under answer-directed updates. The analysis first establishes an ordering result for labels selected by self-vote, then extends the realized-label

parameter	paper text	released code	ours
proposer batch / steps per iter	128 / 5	16 / 6	128 / 5 (= paper)
proposer optimizer updates per iter	20	192	20 (= paper)
solver configured maximum steps per iter	15	20	15 (= paper)
solver global batch	128	128	128
label votes m	10	9 (default, never overridden)	10 (= paper)
difficulty band on plurality fraction	$[0.3, 0.7]$	$[0.3, 0.8]$	$[0.3, 0.8]$ (= code)

Table 13: The paper–code audit. "= paper"/"= code" marks which source each of our settings matches.

update to arbitrary label sources and relates correct-answer probability to pass@k. Fixing the question isolates the effect of the label source.

F.1 Full statements of the main-text results

Fix a training question, let ν be the solver's distribution over final answers, g the correct answer, $p = \nu(g)$, and $B \sim w$ the label used for training. We model one solver update as the KL-regularized optimum for that label, optimizing each question independently:

$$\nu_B^+ = \arg\max_{\rho \ll \nu} \left\{ \rho(B) - \lambda^{-1} \mathrm{KL}(\rho \| \nu) \right\}, \qquad \lambda > 0. \tag{8}$$

The update only redistributes probability among answers the solver already produces ($\rho \ll \nu$). Its optimizer is $\nu_B^+(a) \propto \nu(a) e^{\lambda \mathbf{1}\{a=B\}}$, and for $0 < p < 1$ the expected change in correct-answer probability is

$$\mathbb{E}_B[\nu_B^+(g) - p] = \lambda p(1-p) D + O(\lambda^2), \qquad D = w(g) - \sum_{b \neq g} \frac{\nu(b)}{1-p} w(b). \tag{9}$$

The label source enters only through D, which compares the probability of a correct label with the solver-weighted probability of wrong labels. Every correct label source, including human-annotated labels, has $D = 1$. Appendix F gives the proofs, an exact finite-strength expression, and the assumptions. Equation (12) holds for any label distribution w (Proposition 9); the next proposition states its self-vote case.

Proposition 3 (Self-voted labels reinforce the plurality answer)**.** *Let the label be the plurality of m solver samples, as in R-Zero, with a count-based acceptance band and symmetric tie-breaking. In expectation over the label, the update does not decrease the log-odds of the solver's plurality answer against any other answer. Hence, when the plurality answer is correct, $D \geq 0$ and the expected correct-answer probability does not decrease to first order in λ; when it is wrong, the correct answer does not gain against it. With human-annotated labels, the correct answer instead gains λ in log-odds against every wrong answer, whether or not it is the plurality answer.*

Within this model, a self-voted loop strengthens whatever the solver already ranks first. Correct plurality answers are reinforced, and so are wrong ones. Within this model, changes from a wrong to a correct plurality answer require the sampling noise of individual labels; transfer across questions and finite GRPO steps are outside the model. Human-annotated labels do not have this restriction.

Proposition 4 (Proposer updates reweight questions)**.** *Hold the solver and its labeling and acceptance rule fixed, as in the fixed-student comparison. Let $\mu_\phi(q)$ be the question distribution of proposer ϕ, $\alpha(q)$ the acceptance probability, and $w(b \mid q)$ the label distribution on accepted questions. The proposer affects training only through the distribution of accepted question–label pairs,*

$$\eta_\phi(q, b) = \frac{\mu_\phi(q) \alpha(q) w(b \mid q)}{Z_\phi}, \qquad Z_\phi = \mathbb{E}_{q \sim \mu_\phi}[\alpha(q)] > 0, \tag{10}$$

so the expected first-order gain is $\lambda \, \mathbb{E}_{q \sim \tilde{\mu}_\phi}[p_q(1-p_q) D_q]$ with $\tilde{\mu}_\phi(q) = \sum_b \eta_\phi(q, b)$, where p_q and D_q do not depend on ϕ and each term is at most $\lambda p_q(1-p_q)$. Under self-voted labels, no choice of proposer changes the direction of Proposition 3 on any question.

A better proposer can therefore place more weight on questions whose correct answer is the plurality answer but is not yet certain. It cannot reverse the direction of the update on a question whose plurality answer is wrong, and its gain on these questions shrinks as their answers approach certainty. Within this model, raising the correct answer against a wrong plurality answer requires a signal that does not come from the solver's own answers.

Example: odds versus accuracy. With $m = 10$ votes and the $[0.3, 0.8]$ band, take $\nu = (0.39, 0.40, 0.21)$ for the correct answer g, a wrong answer b, and a third answer. Exact enumeration gives $w \approx (0.440, 0.467, 0.093)$. The log-odds of g against b change by $\lambda(w(g) - w(b)) \approx -0.027\lambda$, while the correct-answer probability changes by about $+0.024\lambda$, because the third answer loses probability.

Proof of Proposition 2. The solver's labeling and acceptance rule are fixed, so conditional on a question q the label distribution $w(\cdot \mid q)$ and the per-question quantities p_q and D_q do not depend on ϕ. A training pair is drawn by sampling $q \sim \mu_\phi$, accepting it with probability $\alpha(q)$, and drawing $b \sim w(\cdot \mid q)$, which gives Eq. (10). The question marginal of η_ϕ is $\tilde{\mu}_\phi(q) = \mu_\phi(q)\alpha(q)/Z_\phi$, and the expected first-order gain is the $\tilde{\mu}_\phi$-average of the per-question gain $\lambda p_q(1 - p_q)D_q$ from Eq. (6). Only the weights $\tilde{\mu}_\phi$ depend on ϕ, so no proposer changes the sign of any per-question term.

F.2 Accepted-experience distribution

For the fixed student θ_5, define the joint voting and acceptance kernel $H_5(b \mid q) = \Pr(K = 1, B = b \mid q, \theta_5)$, where q contains the complete student input. Then $\alpha_5(q) = \sum_b H_5(b \mid q)$ and $w_5(b \mid q) = H_5(b \mid q)/\alpha_5(q)$ wherever $\alpha_5(q) > 0$. Conditioning a proposal from μ_ϕ on acceptance gives

$$\Pr_\phi(q, b \mid K = 1) = \frac{\mu_\phi(q)H_5(b \mid q)}{\sum_{q'} \mu_\phi(q')\alpha_5(q')} = \eta_\phi(q, b).$$

When $\alpha_5(q) = 0$, set $\eta_\phi(q, b) = 0$; the conditional label law is then irrelevant. A proposer with $Z_\phi = 0$ supplies no accepted experience. Under C independent proposals, the expected accepted count is CZ_ϕ. The finite question set's realized size and composition additionally determine which examples are revisited under a fixed update budget.

Proposition 5 (Proposer updates reweight questions)**.** *Fix the student and the kernel H_5, and let $\Delta(q) = \mathbb{E}_{B \sim w_5(\cdot \mid q)}[\nu_B^+(g_q \mid q) - \nu_5(g_q \mid q)]$ be the expected local correctness change on an accepted question under (A1)–(A2). Then for every proposer ϕ with $Z_\phi > 0$, the expected change per accepted example is $\sum_q \eta_\phi(q)\Delta(q)$, with $\eta_\phi(q) = \sum_b \eta_\phi(q, b)$, and to first order*

$$\sum_q \eta_\phi(q)\Delta(q) = \lambda \sum_q \eta_\phi(q)\, p_q(1 - p_q)D_5(q) + O(\lambda^2) \le \lambda \sum_q \eta_\phi(q)\, p_q(1 - p_q) + O(\lambda^2),$$

where $p_q = \nu_5(g_q \mid q) \in (0, 1)$. Under the self-vote hypotheses of Proposition 6, the expected log-odds change of each question's mode against every other supported answer is nonnegative for every ϕ.

Proof. Because $\eta_\phi(q, b) = \eta_\phi(q)w_5(b \mid q)$ and $\Delta(q)$ depends only on $\nu_5(\cdot \mid q)$ and $w_5(\cdot \mid q)$, the expected change is the stated η_ϕ-weighted average. Corollary 10 gives the first-order term, and $D_5(q) \le w_5(g_q \mid q) \le 1$ gives the bound. The log-odds statement is Proposition 9 with $w = w_5(\cdot \mid q)$ combined with Proposition 6; neither depends on ϕ. □

The proposer thus selects which questions receive a fixed per-question update. It cannot change that update's direction, and the bound vanishes as $p_q \to 1$ on the questions it selects.

The proposer intervention holds this kernel fixed. Thus the initial answer law, conditional label law, and $D_5(q)$ (where defined) agree across proposers for the same input, while aggregate label correctness can differ through the question mixture.

F.3 Setup

For question q with correct answer g_q, let $\nu_t(a \mid q) = P_{y \sim \pi_t(\cdot \mid q)}[A(y) = a]$ be the round-t solver's distribution over parsed final answers $a \in \mathcal{A}$ and abstention $\perp$. Probabilities are unconditional; abstentions are not renormalized away. Its committed mode $a_t^*(q) = \arg\max_{a \in \mathcal{A}} \nu_t(a \mid q)$ is assumed unique where used. For self-vote ordering, ν_t is the vote-sampling law, including its prompt, decoding settings, and output budget. Its use as the proximal reference assumes the same answer law; identical model weights alone do not ensure this equality. The correct answer is represented by one reward-equivalent class g_q. The loop draws m samples, forms the plurality $B_{m,t}$, and keeps q iff its plurality fraction falls in the acceptance band; write K for that keep event and

$$w_{m,t}^K(a \mid q) = \Pr\left(B_{m,t} = a \,\middle|\, q, K\right)$$

for the pseudo-label distribution on kept questions. Abstentions do not compete for the plurality, empty votes are rejected, and $w_{m,t}^K(\perp \mid q) = 0$. We distinguish finite-sample voting error from

population-mode error:

$$\varepsilon_{\text{vote}}(m,q) = \Pr\big(B_{m,t} \neq a_t^*(q) \;\big|\; q, K\big), \qquad \varepsilon_{\text{mode}}(q) = \mathbf{1}\{a_t^*(q) \neq g_q\}.$$

The first is finite-sample disagreement with the population mode; the second records whether that mode is correct. Increasing m reduces the corresponding unconditional vote error under standard concentration assumptions, but need not monotonically reduce the error after conditioning on an arbitrary keep event. Two assumptions define the population model. **(A1)** every training question is optimized independently (no parameter sharing); **(A2)** round $t{+}1$ exactly attains the Gibbs optimum of the KL-regularized objective at each question, with KL coefficient $\beta_t > 0$ and update strength $\lambda_t = 1/\beta_t$. These assumptions define an independent-question KL-proximal model of answer reinforcement. Its reference is the current answer distribution at each modeled update. The implemented GRPO trainer instead takes finite parameter updates with group-normalized advantages and a reference frozen within a solver phase. Appendix F.5 distinguishes these update rules; relaxing (A1) additionally introduces cross-question transfer.

F.4 Self-vote selection and update

Proposition 6 (Rank preservation under symmetric plurality selection)**.** *Fix q and t with $\Pr(K \mid q) > 0$, and suppose (i) plurality ties are broken symmetrically among the tied committed answers, and (ii) the keep event K depends only on the committed-answer counts and the abstention count, and is invariant under permutations of committed-answer labels. The agreement band qualifies. Then for all committed answers $a, b \in \mathcal{A}$ and every admissible m, $\nu_t(a \mid q) > \nu_t(b \mid q) \Rightarrow w^K_{m,t}(a \mid q) \geq w^K_{m,t}(b \mid q)$. For a given pair, the inequality is strict if a kept positive-probability count pattern lets one of a, b win while the other is not a maximizer. If this holds for the unique mode against each competitor, $\arg\max_a w^K_{m,t}(a \mid q) = a_t^*(q)$. The strictness condition can fail for a keep rule that accepts only exact ties; the agreement band at the reference configuration (m=10, band $[0.3, 0.8]$) keeps separating patterns and satisfies it.*

This ordering result uses a count-based, permutation-invariant keep rule. The implementation's additional answer-text filters are outside these assumptions.

Lemma 7 (A realized label determines the local pass@k sign)**.** *Fix q, suppress it, and hold one realized label b fixed for the update. With $\lambda = 1/\beta$ and $Z = 1 + \nu(b)(e^\lambda - 1)$, the KL-regularized Gibbs optimum is $\nu'(a) = \nu(a)e^{\lambda \mathbf{1}\{a=b\}}/Z$, so $\nu'(g) = \nu(g)e^\lambda/Z$ if $b = g$ and $\nu(g)/Z$ otherwise, and $\text{pass@}k = 1 - (1 - \nu'(g))^k$. For $0 < \nu(g) < 1$ and $\lambda > 0$, a correct realized label strictly raises every pass@k, and a wrong realized label with $\nu(b) > 0$ strictly lowers every pass@k; at the boundary ($\nu(g) \in \{0, 1\}$ or an unsupported label) pass@k is unchanged, since exact KL regularization cannot create missing support. Here zero support means zero population probability, not absence from a finite set of sampled answers.*

Proposition 8 (The population self-vote update is mode-seeking)**.** *Fix q and adopt (A2) and the hypotheses of Proposition 6. Let the answer reward be expected agreement with a freshly drawn kept pseudo-label, $\bar{r}_t(a \mid q) = w^K_{m,t}(a \mid q)$. Then the KL-regularized optimum is*

$$\nu_{t+1}(a \mid q) \;\propto\; \nu_t(a \mid q)\, \exp\big(\lambda_t\, w^K_{m,t}(a \mid q)\big), \tag{11}$$

and for every supported committed answer $a \neq a_t^*(q)$ the log-odds of the mode against a do not decrease, strictly increasing under the strictness condition of Proposition 6:

$$\log \frac{\nu_{t+1}(a_t^* \mid q)}{\nu_{t+1}(a \mid q)} - \log \frac{\nu_t(a_t^* \mid q)}{\nu_t(a \mid q)} = \lambda_t \big[w^K_{m,t}(a_t^* \mid q) - w^K_{m,t}(a \mid q)\big] \geq 0. \tag{12}$$

Propositions 6 and 8 characterize the optimizer of the expected-label reward. Averaging frozen-label optimizers is a different operation, as considered in Appendix F.5. Neither specifies the finite-training endpoint because m affects realized labels, gradient magnitudes, and the retained-question distribution. Lemma 7 establishes the corresponding local direction of pass@k, without isolating GRPO's negative-advantage term from other optimizer differences.

Mode preservation and aggregate accuracy. Preserving each question's mode does not bound aggregate pass@1 by the initialization's modal accuracy. Consider two equally weighted questions with answer probabilities $(0.40, 0.30, 0.30)$ and $(0.49, 0.51)$, with the correct answer first in each vector. For $m = 10$, keep $3 \leq \max_a N_a \leq 8$, and break vote ties symmetrically. Exact multinomial enumeration gives conditional pseudo-label probabilities $(0.497094, 0.251453, 0.251453)$ and $(0.476659, 0.523341)$. With $\lambda = 5$, Equation 11 raises the first question's correct probability to 0.694818 and lowers the second's to 0.432069. Both modes remain unchanged, yet mean pass@1 is

0.563443, above the initial modal accuracy of 0.5. The initialization's voting accuracy is therefore an empirical comparison reference, not a finite-update bound supplied by these propositions.

Parameter sharing. The implemented trainer shares parameters across questions and therefore violates (A1). Write $M_\theta(q; a, b) = z_\theta(q, a) - z_\theta(q, b)$ for the logit margin between two answer classes at q. A step taken on a different question q' with label $B_{q'}$ changes it by

$$\Delta M_\theta(q; a, b) = -\eta \left\langle \nabla_\theta M_\theta(q; a, b),\, \nabla_\theta \mathcal{L}(q', B_{q'}) \right\rangle + O(\eta^2).$$

The idealized exact-KL update has the log-odds direction derived above; the shared-parameter inner product has no fixed sign. Parameter sharing can therefore change the modal answer, especially when the initial top-two margin is small. The empirical lead-loss analysis records changes of leader after shared-parameter training; it does not isolate the individual cross-question gradient terms.

F.5 Label source and reinforcement direction

The preceding self-vote results use expected agreement with a freshly drawn label as the reward. The implemented loops instead hold one realized label fixed for each question during training. To reflect this distinction, the following result first conditions the update on the realized label and then averages over its distribution. It applies to any label distribution w, including self-vote ($w = w^K_{m,t}$), proposer-stated labels, and human-annotated labels.

Proposition 9 (Label source sets the reinforcement direction)**.** *Fix q and suppress it. Let $B \sim w$ be the round's frozen label and, under (A1)–(A2), let $\nu^+_B(a) = \nu(a)e^{\lambda \mathbf{1}\{a=B\}}/Z_B$ with $Z_B = 1 + \nu(B)(e^\lambda - 1)$ be the KL-regularized optimum for that label (Lemma 7). Then for all answers a, b with $\nu(a), \nu(b) > 0$,*

$$\mathbb{E}_B\left[\log \frac{\nu^+_B(a)}{\nu^+_B(b)}\right] - \log \frac{\nu(a)}{\nu(b)} = \lambda\,[\,w(a) - w(b)\,]. \tag{13}$$

Proof. $\log \nu^+_B(a) - \log \nu^+_B(b) = \log \nu(a) - \log \nu(b) + \lambda(\mathbf{1}\{a = B\} - \mathbf{1}\{b = B\})$, since Z_B cancels in the ratio. Taking the expectation over $B \sim w$ gives $\lambda(w(a) - w(b))$. □

For self-vote labels, averaging the realized-label log-odds change recovers Equation 12. This equality concerns expected log-odds; it does not equate the expected answer probabilities under the two optimization procedures or preserve the leader for every realized label. For human-annotated labels, $w = \delta_g$, and a supported correct answer gains λ in log-odds against every supported wrong answer.

Corollary 10 (First-order change in correct probability)**.** *With $p = \nu(g) \in (0, 1)$ and $D = w(g) - \sum_{b \neq g} \nu(b)\, w(b)/(1 - p)$,*

$$\mathbb{E}_B\big[\nu^+_B(g) - p\big] = \lambda\, p(1 - p)\, D + O(\lambda^2).$$

Every perfectly correct label source has $w(g) = 1$ and $D = 1$, including human-annotated labels. Otherwise, D subtracts the overlap between wrong labels and the solver's own wrong-answer distribution. At $p = 0$ or 1, the exact KL update preserves the boundary; the expression defining D is only used for $0 < p < 1$.

Under the hypotheses of Proposition 6, if g is the unique mode then $D \geq 0$ for $w = w^K_{m,t}$: the weights $\nu(b)/(1 - p)$, $b \neq g$, sum to one (abstention included, with $w(\perp) = 0$), and $w(b) \leq w(g)$ for every committed b.

Proof. For $B = g$, $\nu^+_g(g) = pe^\lambda/(1 + p(e^\lambda - 1)) = p + \lambda p(1 - p) + O(\lambda^2)$. For $B = b \neq g$, $\nu^+_b(g) = p/(1 + \nu(b)(e^\lambda - 1)) = p - \lambda p\, \nu(b) + O(\lambda^2)$. Averaging over w, $\lambda p\,[(1 - p)w(g) - \sum_{b \neq g} \nu(b)w(b)] = \lambda p(1 - p)D$. □

Exact finite-strength comparison. Writing $h = e^\lambda - 1$, the same optimizer gives

$$\mathbb{E}_B[\nu^+_B(g) - p] = hp \left[\frac{(1 - p)w(g)}{1 + hp} - \sum_{b \neq g} \frac{\nu(b)w(b)}{1 + h\nu(b)}\right]. \tag{14}$$

This follows by subtracting p from the correct-label and wrong-label updates before averaging. For a fixed student and correct-label probability, transferring wrong-label mass to an answer with larger $\nu(b)$ reduces this expected gain for every $\lambda > 0$, since $x/(1 + hx)$ is increasing. The sign of D describes the small-strength limit; the exact bracket determines the finite-strength sign.

Equal label accuracy, opposite local directions. Consider a student with answer probabilities $\nu = (0.2, 0.7, 0.1)$, with the correct answer first. Label sources $w_A = (0.4, 0.6, 0)$ and $w_B = (0.4, 0, 0.6)$ are equally accurate, but their wrong labels reinforce the student's common and rare errors, respectively. Their advantages are $D_A = -0.125$ and $D_B = 0.325$, giving local correctness derivatives -0.020 and 0.052. Sufficiently small updates therefore decrease correctness under A and increase it under B. This illustrative calculation concerns the local update and is not an empirical training result. Its derivative need not determine the sign of a finite update: the exact expected correctness under A increases at $\lambda = 1$. The example compares arbitrary label distributions: w_B does not preserve the order of ν and therefore is not a self-vote law under Proposition 6. In the completed proposer comparison, both proposers use the same conditional voting rule; they change which questions are selected.

Update geometry. The criterion D belongs to the KL-proximal objective. Even a tabular softmax policy-gradient step has a different correctness derivative. Let $\nu = \mathrm{softmax}(z)$, let e_b denote a coordinate vector, and take ordinary gradient ascent on the answer-match reward with frozen label b. Its logit gradient is $\nu(b)(e_b - \nu)$. Averaging over labels, the derivative at step size $\eta = 0$ is

$$\frac{d}{d\eta}\mathbb{E}_B \nu_{z+\eta\nu(B)(e_B-\nu)}(g)\bigg|_{\eta=0} = p\sum_b w(b)\nu(b)\left[\mathbf{1}\{b=g\} - p - \nu(b) + \sum_a \nu(a)^2\right].$$

For $\nu = (0.2, 0.7, 0.1)$ and $w = (0.5, 0.5, 0)$, with the correct class first, the KL-proximal derivative is 0.010 but this gradient-step derivative is -0.0024. Thus the sign need not agree even without parameter sharing. Group-normalized advantages, token-level parameterization, and optimizer preconditioning further distinguish implemented GRPO from the KL-proximal solve. A check in our analysis code also enumerates all five-rollout categorical groups and verifies a sign reversal under group normalization. These calculations delimit the simplified model's claims; they do not estimate an experimental training effect. Under independent-question optimization, a held-out question receiving no update is unchanged. Explaining its empirical gain or loss requires cross-question transfer, which this simplified model excludes.

F.6 Initial-answer reinforcement and correction at selected peaks

Table 14 reports the comparisons used in Section 5. Each question's initial leader $a_0(q)$ is its unique most frequent non-abstaining semantic answer among the 128 initial responses. All values are six-model equal-weight means of probability changes (pp, denominator 128) between the initialization and the fixed MathRuler-selected peak, computed from the saved semantic labels. The initially correct group tracks the stable correct class. For the initially wrong group, a wrong answer has no generally stable cross-checkpoint identity; the wrong-answer column is therefore restricted to questions whose initial wrong leader has one strictly parseable exact numeric value in every saved member response, and parse failures are treated as unknown. The correct-answer change on all initially wrong-leading questions needs no such identity match. Selecting the initial leader and estimating its percentage on

Table 14: Initial-answer reinforcement and correction at the selected peaks (pp).

Method	Initial correct answer	Initial wrong numeric answer	Correct answer on all initially wrong questions
R-Zero	$+25.67$	$+16.19$	$+3.09$
Vocabulary Dropout	$+25.02$	$+8.18$	$+2.96$
AZR	$+11.23$	$+3.18$	$+2.19$
SPIRAL	$+19.28$	$+6.11$	$+2.70$
Agent0	$+20.45$	$+8.22$	$+2.87$
Human-annotated RL	$+29.33$	$+7.00$	$+6.89$

disjoint halves of the initial responses gives the same ordering (R-Zero $+16.30$ and human-annotated RL $+8.00$ for the numeric wrong leader). These are descriptive estimates at benchmark-selected peaks: they do not identify the causal effect of the label source, and local reference-label errors in the benchmarks were not repaired.

F.7 Pass@k and correct-answer probability thresholds

The empirical Cover@τ is the fraction of questions whose observed correct-answer frequency is at least τ (Dragoi et al., 2025). Figure 14 reports this complementary diagnostic under rule-based scoring at initialization and each selected peak.

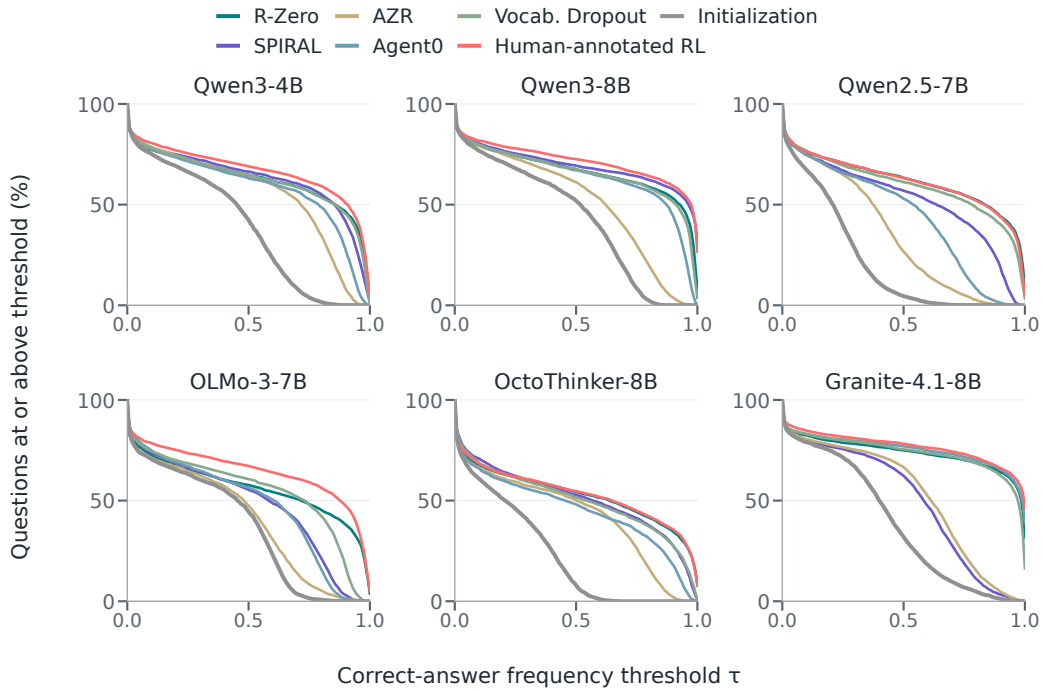

Figure 14: Fraction of questions above each correct-answer probability threshold under rule-based scoring.

Write $C_t(\tau) = \frac{1}{N}\sum_q \mathbf{1}\{p_t(q) \geq \tau\}$ for Cover@τ and $P_k(t) = \frac{1}{N}\sum_q [1-(1-p_t(q))^k]$ for pass@k at the population level. Then

$$P_k(t) = \int_0^1 k(1-\tau)^{k-1}\, C_t(\tau)\, d\tau,$$

because $\int_0^p k(1-\tau)^{k-1} d\tau = 1-(1-p)^k$ for each question; this relation between pass@k and the threshold curve is established by Dragoi et al. (2025) and provides an interpretation of our curves. Pass@1 and pass@128 are therefore two weightings of one correct-answer probability threshold curve, and for any smooth change of the success probabilities

$$\frac{dP_k}{dt} = \frac{1}{N}\sum_q k\,(1-p_t(q))^{k-1}\,\frac{dp_t(q)}{dt}, \tag{15}$$

so pass@1 counts every question's change equally while pass@128 weights a question by $128(1-p)^{127}$, whose relative weight decreases rapidly as p rises. For illustration, if 80% of the questions move from success probability 0.60 to 0.85 and the remaining 20% move from 0.02 to 0.002, pass@1 rises from 48.40% to 68.04% while pass@128 falls from 98.49% to 84.52%. This hypothetical example shows how reliability gains on most questions can coexist with reduced correct-answer sampling probability on a minority of occasionally solved questions. Under the local label update,

$$\frac{d}{d\lambda}\mathbb{E}_B S_k(\nu_B^+(g))\Big|_{\lambda=0} = k\,p(1-p)^k D. \tag{16}$$

The factor $p(1-p)^k$ is maximized at $p = 1/(k+1)$, so the low-success-probability weighting alone does not determine the magnitude of each question's change.

The empirical curve uses $\hat{p}_t(q) = c_t(q)/128$, where $c_t(q)$ is the number of correct responses. Substituting that curve into the integral gives the plug-in estimate $N^{-1}\sum_q[1-(1-c_t(q)/128)^k]$, whereas our reported pass@k uses the unbiased combinatorial estimator in Appendix B.1. They coincide at $k=1$, but not generally: with $c=1$ and $n=k=128$, the plug-in value is about 0.634 and the combinatorial value is 1. The empirical Cover curve is evaluated on the complete $c/128$ grid. Figure 14 uses the complete 129-point frequency grid for every confirmed peak. Table 15 reports both threshold values at the peak and last registered checkpoint for every confirmed setting.

Table 15: Change in reliability-thresholded coverage from initialization (pp).

Model	Recipe	Peak	Peak: $\tau = .1$ / $.5$
Qwen3-4B	R-Zero	r2	+2.20	+22.44
Qwen3-8B	R-Zero	r2	+2.83	+15.70
Qwen2.5-7B	R-Zero	r2	+8.79	+58.93
OLMo-3-7B	R-Zero	r6	+2.02	+13.26
OctoThinker-8B	R-Zero	r3	+7.12	+46.27
Granite-4.1-8B	R-Zero	r1	+3.49	+43.06
Qwen3-4B	SPICE	s100	+1.85	+25.23
Qwen3-8B	SPICE	s100	+3.18	+19.02
Qwen2.5-7B	SPICE	s200	+9.25	+60.01
OLMo-3-7B	SPICE	s100	+2.44	+12.60
OctoThinker-8B	SPICE	s200	-0.17	+6.25
Granite-4.1-8B	SPICE	s200	+3.11	+34.33
Qwen3-4B	SPIRAL	s400	+3.35	+24.04
Qwen3-8B	SPIRAL	s400	+3.77	+17.45
Qwen2.5-7B	SPIRAL	s40	+8.03	+52.34
OLMo-3-7B	SPIRAL	s360	+3.14	+10.89
OctoThinker-8B	SPIRAL	s200	+10.15	+44.84
Granite-4.1-8B	SPIRAL	s320	+1.26	+30.32
Qwen3-4B	AZR	s390	+3.45	+21.56
Qwen3-8B	AZR	s390	+2.65	+9.11
Qwen2.5-7B	AZR	s260	+7.47	+22.16
OLMo-3-7B	AZR	s390	+0.77	+2.97
OctoThinker-8B	AZR	s260	+5.48	+42.08
Granite-4.1-8B	AZR	s390	+1.64	+34.58
Qwen3-4B	Agent0	s100	+2.44	+20.62
Qwen3-8B	Agent0	s90	+2.90	+15.21
Qwen2.5-7B	Agent0	s60	+7.50	+48.53
OLMo-3-7B	Agent0	s100	+4.22	+11.90
OctoThinker-8B	Agent0	s70	+5.20	+40.13
Granite-4.1-8B	Agent0	s80	+4.08	+44.91
Qwen3-4B	Vocab. Dropout	r8	+2.97	+22.44
Qwen3-8B	Vocab. Dropout	r10	+3.11	+15.63
Qwen2.5-7B	Vocab. Dropout	r10	+9.53	+56.73
OLMo-3-7B	Vocab. Dropout	r5	+3.91	+16.19
OctoThinker-8B	Vocab. Dropout	r9	+7.96	+43.48
Granite-4.1-8B	Vocab. Dropout	r4	+4.15	+43.44
Qwen3-4B	Human-annotated RL	s30	+5.55	+26.62
Qwen3-8B	Human-annotated RL	s150	+5.09	+20.66
Qwen2.5-7B	Human-annotated RL	s30	+9.14	+58.86
OLMo-3-7B	Human-annotated RL	s60	+8.06	+22.75
OctoThinker-8B	Human-annotated RL	s150	+7.64	+46.72
Granite-4.1-8B	Human-annotated RL	s105	+5.72	+46.23

F.8 Proofs and finite-sample qualifications

Conventions. $A(y)$ returns the contents of the last `\boxed{}` group in the sampled solution (brace-matched), i.e. the answer the model commits to at the end of its reasoning, or $\perp$ if the sample contains no box; unparsed samples are dropped from vote counts. The mode $a^* = \arg\max_{a \neq \perp} \nu(a)$ is assumed unique with positive mass wherever used; ties require an explicit rule. The implementation keeps the first encountered maximizer; averaging i.i.d. vote order conditional on counts makes this symmetric among tied committed answers. Our Monte Carlo breaks count ties uniformly. $\mathrm{maj}@k$ denotes the accuracy of the plurality returned after parsing k i.i.d. completions, including the specified empty-vote and tie rules; with a unique mode, $\mathrm{maj}@\infty = \mathbb{E}_q \mathbf{1}\{a_q^* = g_q\}$. The loop computes plurality fractions over parsed samples and drops questions with no parsed answer. The trainer's reward is $0.9\, r + 0.1\, r_{\mathrm{fmt}}$; the analysis treats the answer-directed term.

Proof of Proposition 6. Fix q, t, and two committed answers a, b with $\nu_t(a \mid q) > \nu_t(b \mid q)$. If $\nu_t(b \mid q) = 0$, then $w^K_{m,t}(b \mid q) = 0$ and the result is immediate. Otherwise, write $\nu(\cdot) = \nu_t(\cdot \mid q)$ and let $N = (N_c)_{c \in \mathcal{A}}$ be the vector of committed-answer counts among the m samples. Let σ be the involution on count vectors that exchanges the a and b coordinates while leaving the abstention count unchanged. Because the samples are i.i.d.,

$$\frac{\Pr(N = n)}{\Pr(N = \sigma n)} = \left(\frac{\nu(a)}{\nu(b)}\right)^{n_a - n_b},$$

which exceeds 1 exactly when $n_a > n_b$. The pairing argument relies on two properties of the selection rule. First, the keep event is invariant under this swap, so n is kept iff σn is kept. Second, symmetric tie-breaking means the plurality rule commutes with σ: $B_{m,t} = a$ on n iff $B_{m,t} = b$

on σn, with tied outcomes contributing equal probability to both answers. Pairing the two count vectors gives a nonnegative difference: separating pairs have $n_a > n_b$, while tied pairs contribute equally. The difference is strict whenever a kept separating pattern has positive probability. Dividing by $\Pr(K \mid q) > 0$ gives the result. □

Finite-vote error, quantified. For a finite committed-answer set $\mathcal{A}$ with at least two answers, let $a_t^\dagger$ be the committed runner-up. With $\gamma_q = \nu_t(a_t^* \mid q) - \nu_t(a_t^\dagger \mid q)$ the top-two margin, the unconditional version of $\varepsilon_{\text{vote}}$ obeys

$$\begin{aligned}\Pr\big(B_{m,t} \neq a_t^*(q) \mid q\big) &\le \nu_t(\perp \mid q)^m + \sum_{a \in \mathcal{A}\setminus\{a_t^*\}} \exp\Big[-\tfrac{m}{2}\big(\nu_t(a_t^* \mid q) - \nu_t(a \mid q)\big)^2\Big] \\ &\le \nu_t(\perp \mid q)^m + (|\mathcal{A}| - 1)\, e^{-m\gamma_q^2/2}.\end{aligned} \tag{17}$$

where an empty vote returns $\perp$. The first term is its probability; the others follow by applying Hoeffding's inequality to $\sum_{i=1}^m (\mathbf{1}\{A_i = a\} - \mathbf{1}\{A_i = a_t^*\})$ and taking a union bound. This bound can be capped at one. With only one committed answer, the error probability is exactly $\nu_t(\perp \mid q)^m$ and no runner-up margin is defined. The unconditional bound does not constrain $\varepsilon_{\text{mode}}$, which depends only on ν_t and g_q.

Proof of Proposition 8. Under (A2) the optimum of $\max_\pi \mathbb{E}_\pi[\bar{r}_t] - \beta_t \mathrm{KL}(\pi \| \pi_t)$ is the Gibbs distribution $\pi_{t+1}(y) \propto \pi_t(y) e^{\bar{r}_t(A(y)|q)/\beta_t}$; since $\bar{r}_t$ is constant within each answer class, summing within classes gives equation 11. Taking the ratio of equation 11 at a_t^* and at a, the normalizer cancels and the logarithm gives equation 12. The bracket is nonnegative by Proposition 6, and positive under its strictness condition; $\lambda_t > 0$. □

Scope. These results characterize reinforcement under independent, exact KL-regularized updates. The implemented loop uses finite votes, fixes each realized label during training, takes finite GRPO steps, and shares parameters across questions. Cross-question transfer has no fixed sign; consequently, the analysis neither predicts the trained endpoint nor excludes changes in the leading answer. It identifies the systematic reinforcement of currently preferred answers under self-voted labels, providing a local reference for the empirical diagnostics.

G Further Tests

Implications and open tests. The simplified model indicates where an external signal must enter. Labels whose errors do not follow the solver's own errors, such as verified or executable answers, can move a wrong leader; under self-voted labels, a better proposer can only reweight questions. Two interventions would test this under GRPO. The first compares self-voted and verified labels on the same proposer's questions. The second holds valid questions and label accuracy fixed and varies which student errors receive wrong labels. Both remain separate from the completed proposer comparisons. For AZR and SPIRAL, the frozen-source comparisons measure mathematical transfer; execution and game evaluations are needed to assess learning on their native tasks, alongside their original joint-update procedures.