

# Can a Judge Choose Where to Branch?

## Prefix Selection for Tree Rollouts in GRPO on MATH

DC-Jev-RL project, Part III-C

October 2026

### Abstract

Tree rollouts sample a few prefixes per prompt and branch from them, so that the branches share the prefix tokens. A judge that recognises a promising partial solution could choose where to branch. We test this with Qwen3.5-4B-Base on MATH. Offline, Kev-4B, an autoregressive judge of the same size and the policy’s own log-probability all predict across problems whether a 128-token prefix will end correct (AUROC 0.67–0.77), but separate one problem’s prefixes only weakly (0.59–0.64); keeping the two of four prefixes Kev-4B ranks highest raises the share of correct continuations by +4.6 points, the policy’s log-probability by +5.2. In GRPO with tree rollouts (four roots of 128 tokens, two of them continued four times each, 200 steps, three seeds per condition, the third added after the first two had finished), Kev-4B’s choice of roots raised the share of correct leaves by +1.4 points [0.9, 2.0], a random choice by +0.3, and the final policies are as accurate as with random roots (MATH-500: +0.7 [−0.8, 2.3] points). Both tree conditions end above flat GRPO in the seed mean, but that comes mostly from the 2 tree runs that switched to opening their solutions in plain text, a switch that marks the best runs of this project whatever their reward and that none of the 3 flat runs made; the other tree runs end close to flat GRPO. The trees generated 7%–12% fewer tokens than flat sampling of completions of the same lengths. One seed with Kev-4B’s choice came to repeat its solution after the answer, as did one flat seed with no judge at all. A fixed-threshold pruning rule, prune a prefix when the judge’s  $p(\text{yes}) < 0.3$ , prunes 1.6% of the prefixes. Code, logs, every training completion and every number in this paper are in the repository.

## 1 Introduction

Sampling a group of complete solutions per prompt, as GRPO does, spends most of its tokens on solutions that share nothing. Tree-structured rollouts [1] sample a few prefixes and branch from them. This shares the prefix tokens and places the comparison inside the group at the point where solutions diverge. A further step is to choose which prefixes to branch from. A judge that can recognise a promising partial solution could steer the samples towards the parts of the tree where the policy can succeed. That would give GRPO more groups with mixed outcomes, the only groups with a learning signal.

We test both halves of that idea. First, offline: can Kev-4B, the AR judge of the same size, or the policy’s own log-probability tell from the first 128, 256 or 512 tokens whether a solution will end correct? And does keeping the prefixes they rank highest raise the success rate of what is continued? Second, in training: does GRPO with judge-selected prefixes learn better than with randomly selected ones, at the same tree shape and token budget?

## 2 Related work

TreeRL [1] uses on-policy tree search with entropy-guided branching for LLM RL. Process reward models score partial solutions [2, 4]. Math-Shepherd builds its labels by Monte-Carlo continuation, which is how our

prefix study measures a prefix’s value. Search guided by a verifier at test time [3] spends compute where a judge sees promise. Selecting training samples by a learned score changes what the gradient estimates. That is the cost side of the trade-off we measure.

### 3 Judging unfinished solutions

Tree rollouts [1] sample several prefixes of a solution, continue some of them and share each prefix among the continuations. A judge that can tell early which prefixes lead to correct answers could decide which ones to continue. This saves the cost of continuing dead ends and concentrates samples where the policy can succeed. We first measure offline whether a 4B judge can make that call, then train with it.

**Prefix study.** We use GRPO’s validation problems (500 MATH training problems held out from training, §2 of Part II). The base policy samples 4 solutions per problem as in GRPO (temperature 1, the training prompt). Every solution is cut at 128, 256 and 512 tokens if it runs that long. Each prefix gets 4 continuations to the training length (1024 tokens in all), and the verifier checks them. The fraction of a prefix’s continuations that end correct is its Monte-Carlo value. The signals are:

- Kev-4B’s  $p(\text{yes})$  for *on track* (“Is this unfinished solution on track to reach the correct final answer. . . ?”);
- Kev-4B’s  $p(\text{yes})$  for Part II’s *correct* question, with the prefix as the proposed solution;
- the AR judge’s  $p(\text{yes})$  for *on track*;
- the policy’s own mean token log-probability of the prefix, which costs nothing.

For each cut length we report each signal’s AUROC for a continuation ending correct. It is reported pooled and within problems, the comparison that matters when choosing among one problem’s prefixes. We also report each signal’s Spearman correlation with the Monte-Carlo value.

**Pruning experiment.** On the problems whose 4 solutions all run past the cut, a signal keeps the 2 prefixes it ranks highest. We compare the mean value of the kept prefixes with the mean of all 4, which is what a random choice keeps in expectation. We also compare it with an oracle that ranks by half of the continuations and is scored on the other half. There are 95% bootstrap intervals over problems. We also evaluate a fixed-threshold rule: prune a prefix when the judge’s  $p(\text{yes}) < 0.3$ . For it we report how much it prunes, how many pruned prefixes were dead (value 0), and how many correct continuations it throws away.

### 4 Tree GRPO

Each prompt’s group of 8 leaves grows from  $R = 4$  sampled roots of 128 tokens.  $K = 2$  of them are continued, with 4 samples each, from the exact token prefix. `tree_kev` keeps the two roots with the highest Kev-4B *on-track* score; `tree_random` keeps two at random. In distribution, `tree_random` equals sampling two roots only. A root that has already ended is a complete solution, and if kept its 4 leaves are copies of it. Leaves are trained like Part II’s flat completions: the full root and continuation tokens, the verifier’s reward, and the group baseline over the prompt’s 8 leaves. Everything else is Part II’s `r1vr` setup. Every tenth step, the roots that were not kept are continued too, for the log only. This measures what the selection gained on that step against keeping all roots, and against the best two in hindsight.

Two effects pull in opposite directions. Selecting roots by a judge raises the share of correct leaves when the judge is right, which gives GRPO more groups with a learning signal. But it also makes the kept roots no longer samples of the policy, so the gradient is no longer an unbiased estimate of the policy gradient. Comparing `tree_kev` with `tree_random` (same tree shape, unbiased selection) isolates the selection, and comparing `tree_random` with Part II’s `r1vr` isolates the tree shape.

| signal                 | 128 tokens |        |        | 256 tokens |        |        | 512 tokens |        |        |
|------------------------|------------|--------|--------|------------|--------|--------|------------|--------|--------|
|                        | AUROC      | within | $\rho$ | AUROC      | within | $\rho$ | AUROC      | within | $\rho$ |
| Kev-4B <i>on track</i> | 0.76       | 0.61   | 0.53   | 0.77       | 0.64   | 0.52   | 0.77       | 0.65   | 0.46   |
| Kev-4B <i>correct</i>  | 0.70       | 0.59   | 0.40   | 0.75       | 0.59   | 0.48   | 0.77       | 0.58   | 0.47   |
| AR <i>on track</i>     | 0.77       | 0.63   | 0.55   | 0.79       | 0.64   | 0.56   | 0.79       | 0.67   | 0.50   |
| policy log-prob.       | 0.67       | 0.64   | 0.34   | 0.72       | 0.66   | 0.42   | 0.73       | 0.64   | 0.39   |

**Table 1:** How well each signal predicts that a continuation of the prefix ends correct, by prefix length. AUROC pooled over all prefixes and within problems (pairs of continuations of the same problem with different outcomes), and the Spearman correlation  $\rho$  with the prefix’s Monte-Carlo value. 500 validation problems, 4 solutions each, 4 continuations per prefix.

## 5 Can a judge tell a promising prefix?

The base policy’s 500×4 solutions gave 4,515 prefixes and 18,060 verified continuations, of which 37.8% are correct. A prefix’s value spreads over the whole range. At 128 tokens the mean value is 0.41 and 41% of the prefixes are dead (no correct continuation). At 512 tokens these are 0.29 and 61%, because the solutions that are still running by then are the harder ones. So there is something to choose: among one problem’s prefixes, some are worth continuing and some are not.

**All signals carry information, none separates a problem’s prefixes well.** Table 1 gives each signal’s AUROC for a continuation ending correct. Pooled over problems, every signal is clearly informative (0.67–0.77 at 128 tokens). Most of that is the difficulty of the problem, which every signal partly sees. Within problems, the comparison that matters for choosing which prefixes to continue, the AUROCs fall to 0.59–0.64 at 128 tokens and reach at most 0.67 at 512 tokens. Kev-4B’s *on track* question does not do better than the AR judge of the same size asked the same question (within problems 0.64 against 0.64 at 256 tokens, a difference of  $-0.007$  [ $-0.033, 0.020$ ]; 95% intervals from resampling problems). Nor does it beat the policy’s own mean token log-probability, which costs nothing (0.66; Kev-4B minus log-probability  $-0.021$  [ $-0.055, 0.014$ ]). Pooled over problems both judges are ahead of the log-probability ( $+0.053$  [ $0.026, 0.080$ ] for Kev-4B), so what the judges add is mostly a sense of how hard the problem is, which does not help to choose among one problem’s prefixes. Asking Kev-4B Part II’s *correct* question about an unfinished solution works worse than asking whether it is on track.

**Selecting prefixes helps a little, and any signal will do.** On the problems whose four solutions all run past the cut, keeping the two prefixes a signal ranks highest raises the mean value of what is continued (Table 2). At 128 tokens Kev-4B’s *on track* gains  $+0.046$  [ $0.034, 0.058$ ] over the 0.39 of keeping all four. The AR judge gains  $+0.045$  and the policy’s log-probability  $+0.052$ . The best gains are at 256 tokens,  $+0.064$  for Kev-4B,  $+0.069$  for the AR judge and  $+0.065$  for the log-probability. The oracle, which ranks by two of the four continuations and is scored on the other two, gains  $+0.098$  there. With four continuations per prefix the value itself is a noisy estimate, so the oracle’s gain underestimates what perfect knowledge would give. At 128 tokens the signals gain about as much as this oracle, at 256 tokens about two thirds of it. In absolute terms the gains are small: a few more correct continuations in a hundred.

**A fixed pruning threshold rarely fires.** The rule prunes a prefix when the judge’s  $p(\text{yes})$  is below 0.3 (Table 3). Kev-4B’s *on track* probability is almost never that low. The rule prunes 1.6% of the prefixes at 128 tokens. 84% of those are dead, but they are only 3% of all dead prefixes. With the *correct* question the rule prunes more (12.1%), with fewer dead prefixes among the pruned (61%), and it throws away 6.4% of

| keep 2 of 4 by         | 128 tokens            | 256 tokens            | 512 tokens             |
|------------------------|-----------------------|-----------------------|------------------------|
| problems               | 401                   | 253                   | 93                     |
| value of all 4         | 0.386                 | 0.324                 | 0.165                  |
| Kev-4B <i>on track</i> | +0.046 [0.034, 0.058] | +0.064 [0.047, 0.082] | +0.040 [0.015, 0.067]  |
| Kev-4B <i>correct</i>  | +0.040 [0.028, 0.052] | +0.043 [0.026, 0.060] | +0.021 [−0.003, 0.046] |
| AR <i>on track</i>     | +0.045 [0.033, 0.057] | +0.069 [0.053, 0.085] | +0.048 [0.022, 0.075]  |
| policy log-prob.       | +0.052 [0.041, 0.063] | +0.065 [0.048, 0.083] | +0.040 [0.014, 0.067]  |
| oracle (split half)    | +0.048 [0.033, 0.063] | +0.098 [0.080, 0.117] | +0.059 [0.034, 0.086]  |

**Table 2:** Mean value of the 2 prefixes (of 4) a signal keeps, minus the mean value of all 4, on the problems whose 4 solutions all run past the cut. 95% bootstrap intervals over problems. The oracle ranks by half of the continuations and is scored on the other half.

|                        | 128 tokens |      |      | 256 tokens |      |      | 512 tokens |      |      |
|------------------------|------------|------|------|------------|------|------|------------|------|------|
| signal                 | pruned     | dead | lost | pruned     | dead | lost | pruned     | dead | lost |
| Kev-4B <i>on track</i> | 1.6%       | 84%  | 0.2% | 2.1%       | 85%  | 0.3% | 1.5%       | 87%  | 0.3% |
| Kev-4B <i>correct</i>  | 12.1%      | 61%  | 6.4% | 12.1%      | 72%  | 4.2% | 17.1%      | 76%  | 8.2% |
| AR <i>on track</i>     | 4.6%       | 77%  | 1.2% | 8.0%       | 78%  | 1.7% | 14.4%      | 87%  | 3.9% |

**Table 3:** The threshold rule, prune when  $p(\text{yes}) < 0.3$ : the fraction of prefixes pruned, the fraction of the pruned that were dead, and the fraction of all correct continuations that the pruned prefixes would have produced.

all correct continuations. A fixed threshold on an uncalibrated probability is not a usable pruning rule here; ranking within a problem is.

## 6 Training with tree rollouts

Both tree conditions ran with 3 seeds of 200 steps, everything else as in Part II’s `rlvr`, which is the flat reference (3 seeds). Runs with the same seed see the same problems in the same order. We first ran two seeds of each tree condition. Each condition then had one seed far above `rlvr` and one at its level, which left the effect of the tree shape undecided, so after both had finished we added a third seed of each (`scripts/make_jobs.sh`). That decision was made after seeing the first results; the third seeds are reported like the others. Table 5 gives the final accuracies, Table 6 those of each seed, Table 7 the paired differences, Table 4 what happened inside the trees during training, and Figure 1 the training curves. The intervals resample problems and do not include the variation between seeds.

### 6.1 The judge’s choice of roots

**The judge’s choice gained little over a random one.** Every tenth step the roots that were not kept are continued too, so we know what keeping the chosen two gained over keeping all four (Table 4). Under `tree_kev` the leaves of the two roots Kev-4B ranked highest were correct +1.4 [0.9, 2.0] points more often than the leaves of all four roots, over 60 audited steps. Under `tree_random`, whose choice gains nothing in expectation, the same number is +0.3 [−0.2, 0.8]; these intervals resample the audited steps. Keeping the two roots whose leaves turned out best would have gained +8.4 points, more than any ranking made before the leaves exist could, since the same four leaves per root both rank the roots and score them. On the audited steps Kev-4B’s *on track* score of a root separates its correct from its incorrect leaves with an AUROC of 0.75 pooled over prompts, but only 0.55 [0.53, 0.57] within a prompt, the comparison that decides which roots are

| condition   | seeds | roots | audit, every 10th step: all four roots continued |               |       |              |              | tokens       |
|-------------|-------|-------|--------------------------------------------------|---------------|-------|--------------|--------------|--------------|
|             |       | ended | AUROC                                            | within prompt | value | gain, kept 2 | gain, best 2 | per step (k) |
| tree_random | 3     | 1.5%  | 0.75                                             | 0.56          | 60.6  | +0.3         | +8.1         | 65.1         |
| tree_kev    | 3     | 1.3%  | 0.75                                             | 0.55          | 59.9  | +1.4         | +8.4         | 74.7         |

**Table 4:** Inside the trees during training (mean over steps and seeds): share of roots that had already ended within 128 tokens; on the audited steps (every tenth), where all four roots of a prompt are continued, the AUROC of Kev-4B’s *on track* score of a root for its leaves’ verdicts, pooled and within prompt, the leaf accuracy of all four roots (percent), and what keeping the chosen two, or the best two in hindsight, gained over it (points); tokens generated per step (thousands, last 20 steps), not counting the audit’s continuations.

| condition   | seeds | MATH-500   | GSM8K      | AMC23      | AIME24     | AIME25     |
|-------------|-------|------------|------------|------------|------------|------------|
| base model  | –     | 76.5       | 87.9       | 50.6       | 10.0       | 10.9       |
| r1vr        | 3     | 82.5 ± 0.6 | 93.0 ± 0.3 | 60.9 ± 2.4 | 21.5 ± 0.4 | 16.7 ± 2.1 |
| tree_random | 3     | 84.1 ± 2.2 | 93.3 ± 0.8 | 65.7 ± 4.2 | 25.6 ± 6.8 | 19.9 ± 4.4 |
| tree_kev    | 3     | 84.8 ± 2.3 | 93.9 ± 0.7 | 66.0 ± 2.7 | 25.2 ± 5.9 | 19.6 ± 4.4 |

**Table 5:** Final policies: accuracy in percent, seed mean ± standard deviation over seeds. MATH-500 and GSM8K: greedy. AMC23, AIME24 and AIME25: mean over 16 samples per problem at temperature 0.6. r1vr: flat sampling (Part II).

kept (0.56 under `tree_random`). Offline, with the same structure (four prefixes of 128 tokens per problem, four continuations each), the within-problem AUROC was 0.61, and keeping the two best-ranked prefixes raised the share of continuations ending correct by +4.6 points (§5). The prefixes differ: offline they are the base policy’s, in training the policy’s own as it learns. Its roots lead to correct answers more and more often (under `tree_random`, 54.5% of the audited leaves were correct in the first half of training and 66.7% in the second, against 38.6% of the continuations of the base policy’s prefixes offline), and the room for a choice, the best two in hindsight, shrank from +9.6 to +6.7 points.

**Final accuracy: no effect of the judge’s choice.** With the same tree shape, keeping Kev-4B’s roots instead of random ones changed the final accuracy by +0.7 [−0.8, 2.3] points on MATH-500, +0.6 [0.0, 1.2] on GSM8K, +0.4 [−1.7, 2.6] on AMC23, −0.3 [−3.7, 2.6] on AIME24 and −0.3 [−2.6, 1.9] on AIME25 (Table 7). No interval excludes zero; the one on GSM8K just reaches it.

**Against flat sampling, the seed means follow the seeds that switched.** In the seed mean both tree conditions end above r1vr, by +1.5 and +2.3 points on MATH-500 and by +4.0 and +3.7 on AIME24. Seed by seed the picture is different (Table 6). 2 of the tree runs switched to opening their solutions in plain text (§6.2), and they end at 86.6–87.2% on MATH-500 and 31.9–33.3% on AIME24. The other 4 end at 82.4–84.6% and 20.8–22.9%, and the 3 seeds of r1vr, none of which switched, at 81.8–83.0% and 21.0–21.9%. The seed means of a condition thus mostly count how many of its seeds switched. Whether the tree shape helps a policy that does not switch, or makes the switch more likely, these runs cannot tell. At step 100, when no run drafted in more than 5% of its MATH-500 completions, neither tree condition was ahead on MATH-500 (81.1% and 81.9%, against 82.4% for r1vr).

| condition   | seed | MATH-500 | GSM8K | AMC23 | AIME24 | AIME25 |
|-------------|------|----------|-------|-------|--------|--------|
| rlvr        | 1    | 82.8     | 93.0  | 63.6  | 21.0   | 18.8   |
|             | 2    | 83.0     | 93.3  | 59.8  | 21.9   | 16.9   |
|             | 3    | 81.8     | 92.6  | 59.2  | 21.7   | 14.6   |
| tree_random | 1    | 82.4     | 92.4  | 63.4  | 21.0   | 16.2   |
|             | 2    | 86.6     | 93.8  | 70.5  | 33.3   | 24.8   |
|             | 3    | 83.2     | 93.7  | 63.1  | 22.3   | 18.8   |
| tree_kev    | 1    | 87.2     | 94.7  | 68.4  | 31.9   | 24.4   |
|             | 2    | 82.6     | 93.5  | 63.1  | 20.8   | 15.6   |
|             | 3    | 84.6     | 93.6  | 66.6  | 22.9   | 18.8   |

**Table 6:** The final accuracy of each seed, in percent. The third seed of each tree condition was added after the first two had finished.

| minus tree_random | MATH-500    | GSM8K       | AMC23        | AIME24       | AIME25       |
|-------------------|-------------|-------------|--------------|--------------|--------------|
| rlvr              | -1.5        | -0.3        | -4.8         | -4.0         | -3.2         |
|                   | [-3.1, 0.0] | [-1.0, 0.4] | [-8.3, -1.7] | [-7.6, -0.8] | [-5.6, -1.1] |
| tree_kev          | +0.7        | +0.6        | +0.4         | -0.3         | -0.3         |
|                   | [-0.8, 2.3] | [0.0, 1.2]  | [-1.7, 2.6]  | [-3.7, 2.6]  | [-2.6, 1.9]  |

**Table 7:** Paired differences to `tree_random` in percentage points: per problem, each condition’s seed mean, averaged over problems, with a 95% bootstrap interval over problems. `tree_kev` minus `tree_random` isolates the judge’s choice of roots; `rlvr` minus `tree_random` the tree shape.

## 6.2 Opening in plain text

Part II found that some GRPO runs switch, abruptly, to opening their completions with an informal draft in plain text, without LaTeX or markdown, and that the runs that switch are the best ones (Part II, §8.3). We measure the style as there, by the share of completions that open with such a draft: no math delimiter, markdown heading or bold in the first 300 characters, a measure defined after reading the completions.

**One seed of each tree condition switched.** The base model opens 11% to 12% of its MATH-500 completions this way, and every run cut that share in its first 50 steps, to at most 8% at step 50 (Table 8). In `tree_random` seed 2 it then rose from 2% at step 100 to 83% at step 150, and in `tree_kev` seed 1 from 4% at step 150 to 78% at step 200. Every other run, the 3 of `rlvr` included, opens at most 7% of its final completions this way. The two runs that switched write differently from those of Part II: in most of their completions the plain text is the whole solution, and only 25% and 15% of their drafts go on to formatted text, against 85% and 58% in the runs of Part II that switched (`rlvr_kev_gate` seed 2 and `rlvr_kev_add` seed 3).

In a tree, how a completion opens is settled by its root. The four leaves of a root share its first 128 tokens, which nearly always hold the first 300 characters: they agree on whether they open with a draft for all but at most 0.3% of the roots of a run. Keeping two roots of four is thus also a choice among openings, made by Kev-4B under `tree_kev` and at random under `tree_random`; the random choice did not keep a run from switching.

**The verdicts do not show why.** As in Part II, we compare within each group of training completions that holds both kinds how often the drafts and the others are right (Table 9); in a tree run such a group has one

| condition   | seed | MATH-500 | AIME24 | share of drafts, step |    |     |     |     | final |
|-------------|------|----------|--------|-----------------------|----|-----|-----|-----|-------|
|             |      |          |        | 0                     | 50 | 100 | 150 | 200 |       |
| rlvr        | 1    | 82.8     | 21.0   | 12%                   | 1% | 0%  | 1%  | 1%  | 1%    |
|             | 2    | 83.0     | 21.9   | 12%                   | 0% | 0%  | 0%  | 0%  | 0%    |
|             | 3    | 81.8     | 21.7   | 12%                   | 1% | 1%  | 0%  | 1%  | 1%    |
| tree_random | 1    | 82.4     | 21.0   | 12%                   | 1% | 1%  | 1%  | 3%  | 3%    |
|             | 2    | 86.6     | 33.3   | 11%                   | 1% | 2%  | 83% | 83% | 82%   |
|             | 3    | 83.2     | 22.3   | 12%                   | 6% | 2%  | 1%  | 2%  | 2%    |
| tree_kev    | 1    | 87.2     | 31.9   | 11%                   | 3% | 5%  | 4%  | 78% | 78%   |
|             | 2    | 82.6     | 20.8   | 12%                   | 4% | 2%  | 1%  | 3%  | 3%    |
|             | 3    | 84.6     | 22.9   | 12%                   | 8% | 4%  | 4%  | 6%  | 7%    |

**Table 8:** The share of MATH-500 completions that open with an informal draft (no math delimiter, markdown heading or bold in the first 300 characters), at each periodic evaluation (greedy, every 50 steps) and in the final evaluation, per run, with the final accuracies.

| condition   | seed | steps 0–49 |                | steps 50–99 |                | steps 100–149 |                | steps 150–199 |                |
|-------------|------|------------|----------------|-------------|----------------|---------------|----------------|---------------|----------------|
|             |      | drafts     | $\Delta$ right | drafts      | $\Delta$ right | drafts        | $\Delta$ right | drafts        | $\Delta$ right |
| rlvr        | 1    | 7%         | −8.5           | 2%          | +4.2           | 1%            | −3.9           | 2%            | +1.8           |
|             | 2    | 5%         | −15.3          | 1%          | −9.3           | 0%            | −5.0           | 0%            | −2.1           |
|             | 3    | 6%         | −14.3          | 1%          | −0.7           | 1%            | +3.3           | 1%            | +2.6           |
| tree_random | 1    | 9%         | −1.4           | 2%          | −10.7          | 2%            | +4.2           | 2%            | +11.4          |
|             | 2    | 10%        | −9.8           | 2%          | −18.8          | 40%           | +1.2           | 75%           | +0.4           |
|             | 3    | 13%        | −2.2           | 5%          | −3.3           | 2%            | +9.4           | 1%            | +3.8           |
| tree_kev    | 1    | 9%         | −4.4           | 5%          | −5.6           | 8%            | −5.2           | 20%           | +1.7           |
|             | 2    | 12%        | −5.4           | 3%          | +1.3           | 2%            | +0.0           | 2%            | −0.9           |
|             | 3    | 12%        | −3.8           | 7%          | −0.5           | 5%            | −2.7           | 6%            | +3.6           |

**Table 9:** Training completions that open with an informal draft, per 50-step window: their share, and  $\Delta$  right, the mean over the groups that hold both kinds of the share of drafts the verifier accepts minus that of the other completions of the group, in percentage points.

root of each kind. In steps 0–49 drafts were less often right than the others in all 9 runs (the difference was between −15.3 and −1.4 points), which is why their share fell. In the windows in which they spread, they were about as often right as the others: +1.2 points in `tree_random` seed 2 over steps 100–149, when 40% of its training completions were drafts, and +1.7 in `tree_kev` seed 1 over steps 150–199 (20%). What made these runs switch, the verdicts we logged do not show. Across the 26 GRPO runs of this project, the 6 that switched, in 5 conditions and 2 of them with the verifier’s reward alone, are the 6 best on AIME24 (30.4 to 34.6%, every other run at most 24.4%; Part II, §8.3).

### 6.3 Repetition

**One seed of `tree_kev` came to repeat itself, as one of `rlvr` did.** The training completions of `tree_kev`’s seed 2 came to repeat part of the solution after the first answer: 4% of them did in steps 0–24, 11% in steps 50–74, 41% in steps 100–124, 65% in steps 150–174 and 59% in steps 175–199 (Table 10; the share first exceeded a fifth in the window from step 75). They grew from 620 to 859 tokens on average; in the last 25 steps 51% of them were cut off at the length limit of 1024 tokens and the verifier accepted 63%. The final

| condition          | seed | training completions, steps |       |         |         |         | final    |
|--------------------|------|-----------------------------|-------|---------|---------|---------|----------|
|                    |      | 0–24                        | 50–74 | 100–124 | 150–174 | 175–199 | MATH-500 |
| <b>rlvr</b>        | 1    | 5%                          | 0%    | 0%      | 0%      | 1%      | 0%       |
| <b>rlvr</b>        | 2    | 3%                          | 0%    | 0%      | 0%      | 0%      | 0%       |
| <b>rlvr</b>        | 3    | 5%                          | 19%   | 65%     | 74%     | 75%     | 90%      |
| <b>tree_random</b> | 1    | 3%                          | 1%    | 0%      | 0%      | 0%      | 0%       |
| <b>tree_random</b> | 2    | 6%                          | 2%    | 7%      | 0%      | 0%      | 0%       |
| <b>tree_random</b> | 3    | 3%                          | 2%    | 0%      | 0%      | 0%      | 0%       |
| <b>tree_kev</b>    | 1    | 3%                          | 0%    | 0%      | 0%      | 0%      | 0%       |
| <b>tree_kev</b>    | 2    | 4%                          | 11%   | 41%     | 65%     | 59%     | 90%      |
| <b>tree_kev</b>    | 3    | 2%                          | 4%    | 3%      | 0%      | 0%      | 0%       |

**Table 10:** Share of completions that repeat part of the solution after the first boxed answer (a line of at least 20 characters that already occurred before it): the training completions of each run in five 25-step windows, and the final policy’s greedy MATH-500 completions.

| condition          | seed | tokens | truncated | repeated lines |              | several | draft   | Kev-4B $P_c$ |       |
|--------------------|------|--------|-----------|----------------|--------------|---------|---------|--------------|-------|
|                    |      |        |           | share          | after answer | answers | opening | right        | wrong |
| <b>rlvr</b>        | 1    | 541    | 2%        | 1%             | 0%           | 2%      | 1%      | 0.84         | 0.60  |
|                    | 2    | 555    | 4%        | 1%             | 0%           | 6%      | 0%      | 0.85         | 0.59  |
|                    | 3    | 909    | 8%        | 28%            | 90%          | 95%     | 1%      | 0.88         | 0.64  |
| <b>tree_random</b> | 1    | 553    | 3%        | 1%             | 0%           | 3%      | 3%      | 0.83         | 0.60  |
|                    | 2    | 549    | 7%        | 0%             | 0%           | 46%     | 82%     | 0.88         | 0.56  |
|                    | 3    | 582    | 4%        | 1%             | 0%           | 5%      | 2%      | 0.84         | 0.57  |
| <b>tree_kev</b>    | 1    | 567    | 8%        | 0%             | 0%           | 18%     | 78%     | 0.88         | 0.55  |
|                    | 2    | 931    | 7%        | 24%            | 90%          | 94%     | 3%      | 0.89         | 0.66  |
|                    | 3    | 485    | 2%        | 1%             | 0%           | 1%      | 7%      | 0.83         | 0.57  |

**Table 11:** What each final policy writes on MATH-500 (greedy): completion tokens, share truncated at 2048 tokens, share of lines (of at least 20 characters) that repeat an earlier line of the same completion, share of completions that repeat such a line after their first boxed answer, share with more than one boxed answer (the verifier reads the last one), share that open with an informal draft, and Kev-4B’s  $P_c$  for right and for wrong answers.

policy repeats after its first answer on 90% of the MATH-500 problems, writes 931 tokens per solution, and its accuracy, 82.6%, is that of **rlvr**. Seed 3 of **rlvr** did the same (from the window at step 75; its final policy repeats on 90% of the problems). In every other run at most 7% of the training completions of any window from step 50 on repeated after the answer. More than one boxed answer is not repetition: **tree\_random** seed 2 gives several on 46% of the MATH-500 problems while 0.2% of its lines repeat an earlier one, against 24.4% for seed 2 of **tree\_kev**.

Both runs that repeat are rewarded by the verifier alone, which reads the last answer, so a repetition that restates the answer costs them nothing. Kev-4B would have paid for it, had it scored whole solutions: on the final MATH-500 solutions of **tree\_kev** seed 2 that repeat lines, its  $P_c$  is +0.046 [0.041, 0.051] higher as written than cut after the first answer on the 348 the verifier accepts, and +0.085 [0.068, 0.103] higher on the 54 it rejects (the probe of Part II, §8.4). Under **tree\_kev**, however, it scores only the roots, the first 128 tokens, long before any answer, so its choice cannot have paid for the repetition, and **rlvr** seed 3 repeats with no judge at all.

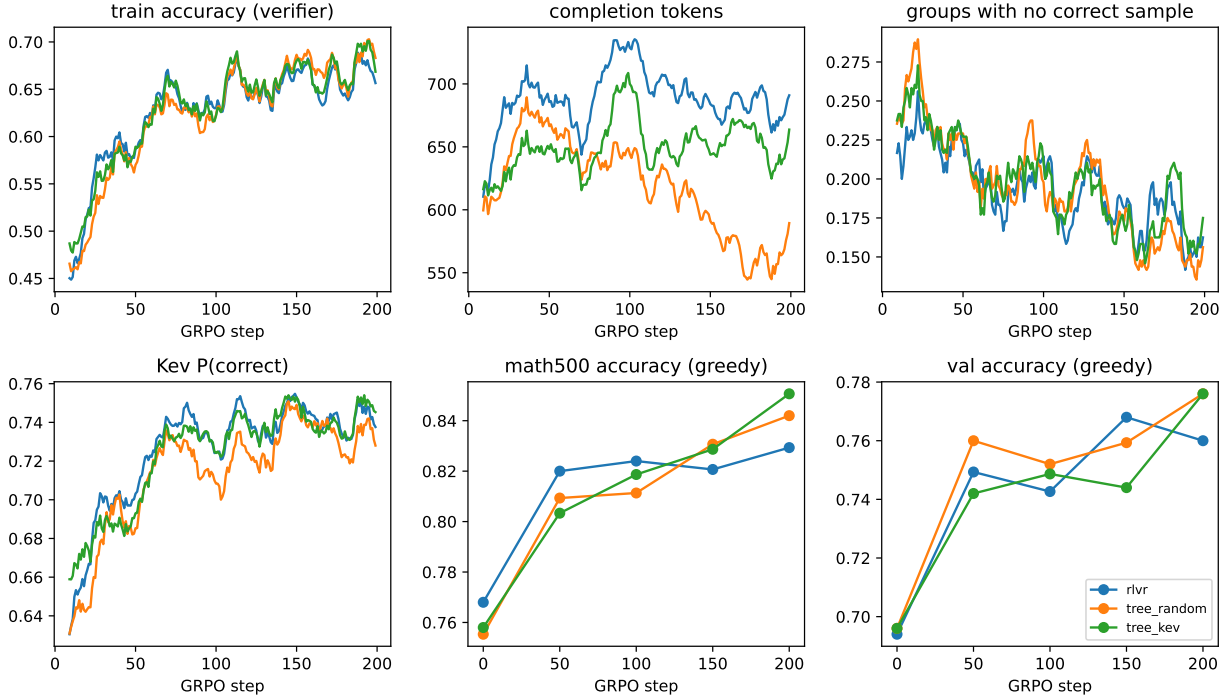


**Figure 1:** Training curves (seed mean): verifier accuracy, completion tokens, share of groups with no correct completion and Kev-4B’s  $P_c$  on the training completions (10-step moving averages), and greedy accuracy on MATH-500 and the validation set every 50 steps.

## 6.4 Cost and evaluation noise

**Cost.** A tree samples four roots of at most 128 tokens per prompt and continues two of them four times each, so the leaves of a root share its tokens. In the last 20 steps the tree runs generated 11% to 12% (`tree_random`) and 7% to 12% (`tree_kev`) fewer tokens than flat sampling of completions of the same lengths would have. Per step, `tree_random` generated 65.1 thousand tokens and `tree_kev` 74.7 thousand, against 87.0 thousand for `rlvr`; these means also depend on how long each seed’s completions grew (`rlvr` seed 3 and `tree_kev` seed 2, which repeat themselves, generated 108.9 and 101.6 thousand). Scoring the roots with Kev-4B on the shared backbone is the only extra cost of the judge’s choice.

**Evaluation noise.** Across the 9 runs in this comparison, the base model’s MATH-500 accuracy at step 0 ranges from 75.2% to 78.0%, and a run’s accuracy changes by 1.4 points on average between its evaluations at steps 150 and 200. The final weights of each run are evaluated twice on MATH-500, both times greedily: at step 200 with the periodic evaluations, and in the final evaluation. The two results differ by 0.6 points on average and by up to 1.4 (9 runs). A difference of about a point between two single evaluations is within the noise of the evaluation itself.

## 7 Discussion

**Within a problem, the judge sees little of where a solution is going.** Offline, Kev-4B’s *on track* score separates the correct from the incorrect continuations of one problem’s 128-token prefixes with an AUROC of 0.61, no better than the AR judge of the same size (0.63) or the policy’s own log-probability (0.64), which costs nothing. Across problems every signal does better, because each partly sees how hard the problem is,

and that does not help to choose among one problem’s prefixes. In training, on the policy’s own roots, the separation was weaker still (0.55 [0.53, 0.57]), and keeping Kev-4B’s two roots of four raised the share of correct leaves by +1.4 [0.9, 2.0] points, against +0.3 for a random choice and +8.4 for the best two in hindsight. The gain was larger early in training (+2.2 points in the first half, +0.7 in the second, without intervals), while the policy’s roots came to lead to correct answers more often and the room for a choice shrank (best two in hindsight: +9.8 and +7.0 points).

**A small gain at the roots did not change what the policy learned.** With the same tree shape, keeping Kev-4B’s roots instead of random ones left the final accuracy unchanged within the intervals over problems on all five benchmarks; on GSM8K the difference, +0.6 [0.0, 1.2] points, has an interval that just reaches zero. The idea behind the judge’s choice is that it gives GRPO more groups with mixed outcomes, the only groups with a learning signal. It did not: in the last 20 steps 17% of the groups had no correct leaf under `tree_kev` and 15% under `tree_random`, and 48% and 47% had only correct ones. The choice has a cost in principle: the kept roots are no longer samples of the policy, so the gradient is no longer an unbiased estimate of the policy gradient. We saw no effect of that on accuracy. Nor did the judge’s choice make the policy repeat itself: one seed of `tree_kev` came to repeat its solution after the first answer, but so did one seed of `rlvr`, which has no judge at all, and Kev-4B scores only the first 128 tokens of a solution, long before the answer.

**The tree shape: the seeds that switched decide the means.** Both tree conditions end above flat sampling in the seed mean, by +1.5 and +2.3 points on MATH-500 and +4.0 and +3.7 on AIME24, but most of that comes from the 2 tree runs that switched to opening their solutions in plain text. These end at 31.9–33.3% on AIME24, the others at 20.8–22.9% and `rlvr` at 21.0–21.9%. The same switch marks the best runs of the whole project, whatever their reward (Part II, §8.3). Here 2 of the tree runs made it and none of `rlvr`’s 3 seeds. Whether trees make it more likely, so few runs cannot tell, the less so as we did not plan for the switch, found it after the runs had finished, and classify the runs by a measure defined after reading their completions. In a tree the root settles how a completion opens, so the choice of roots is also a choice among openings; a random choice of roots was enough for a run to switch. What the tree shape does buy for certain is tokens: the trees generated 7%–12% fewer tokens per step than flat sampling of completions of the same lengths would have.

**Limitations.** One policy size (4B), one judge question (*on track*) asked of 128-token roots, one tree shape (four roots, two kept, four leaves each), 200 steps at the learning rate of Part II, and 3 seeds per tree condition (`rlvr`: 3), the third of each added after the first two had finished. With more roots per prompt a judge has more to choose from, and so do its errors. The audit continues all roots only every tenth step, 20 audited steps per run; its intervals resample those steps, not seeds. The intervals of the final accuracies resample problems, not seeds. The offline study used the base policy’s prefixes, not the training policy’s. The style measures were defined after reading the completions. We report tokens, not time: the runs shared GPUs with other experiments.

## Reproducibility

Code, prompts, raw logs (every GRPO step and evaluation), every training completion of the GRPO runs, and the prefix study’s solutions, prefixes, continuations and judge scores are in the repository (`experiments/part3c_tree/`). The tables and every number in the text are generated from them by `dcjev/llm/prefix_study.py`, `dcjev/analysis/part2.py` and `dcjev/analysis/paper_numbers.py`.

## References

- [1] Zhenyu Hou, Ziniu Hu, Yujiang Li, Rui Lu, Jie Tang, and Yuxiao Dong. TreeRL: LLM reinforcement learning with on-policy tree search. In *Annual Meeting of the Association for Computational Linguistics*, pages 12355–12369, 2025.
- [2] 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.
- [3] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. In *International Conference on Learning Representations*, 2025.
- [4] Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Annual Meeting of the Association for Computational Linguistics*, 2024.