When RL Suppresses Its Own Vocabulary: Recovering Reasoning Diversity in Puzzle-to-Math Transfer

Mayug Maniparambil [1] **Arjun Karuvally** [2] **Terrence Sejnowski** [2] **Fergal Reid** [1]

Abstract

Reinforcement learning using verifiable rewards (RLVR) improves LLM reasoning, but the conditions under which it transfers across domains—and why it does so—remain under-explored. We study cross-domain transfer in a 7B model whose SFT and RL post-training stages use only constraint-satisfaction puzzles, with no mathematics problems in the post-training data. To analyze how transfer emerges, we introduce a reasoning primitive-level framework that combines a 9-class span classifier with motif extraction, allowing us to segment chain-of-thought traces into primitive motifs and track their evolution across training stages and domains. We find that puzzle SFT induces a reasoning-primitive vocabulary, yielding a +7pp `pass@32` gain on OlymMATH-Hard. Vanilla GSPO then composes these primitives into longer compute–verify chains, adding a further +6pp. However, this RL stage also suppresses exploratory primitives such as *hypothesize* and *backtrack*. To address this, we introduce a novelty bonus that rewards diverse correct rollouts, using perplexity under the reference model as a signal. This partially recovers these primitives during RL and adds a further +7pp `pass@32` relative to vanilla GSPO. Finally, the end-to-end recipe raises the hard-math capability ceiling from 16.0% at the OLMo3-7B-Instruct-SFT base to 36.0%, without adding any mathematics problems during the SFT or RL stages.

1. Introduction

Reinforcement learning has driven much of the recent progress in large language model (LLM) reasoning on math and code, but how much of this improvement reflects general problem-solving capability versus skills specific to the training distribution is unclear. We probe this in a setting designed to dissociate the two: a 7B model is trained with supervised fine-tuning (SFT) and reinforcement learning (RL) on constraint-satisfaction puzzles only — no mathematics problems appear in post-training — and is evaluated on hard mathematics benchmarks where current models remain far from saturation. Puzzles and mathematics share problem-solving operations such as search, planning, verification, and backtracking (Pólya, 1945; Simon & Newell, 1971; Schoenfeld, 2014), while differing substantially in problem framing and notation. Whatever RL contributes to math performance in this setup cannot come from post-training exposure to mathematics. If transfer occurs, it must reflect changes to the abstract operations the two domains share. This setup lets us ask: does RL on puzzles improve hard-mathematics performance, and if so, what reasoning-level changes are responsible?

Transfer to math occurs: after SFT and RL on puzzles, `pass@32` on OlymMATH-Hard rises by 20 percentage points (from 16.0% to 36.0%) without any mathematics in post-training. To understand what reasoning-level changes drive this gain, we develop a primitive-level analysis framework. A span classifier segments chain-of-thought reasoning into 9 primitives — HYPOTHESIZE, COMPUTE, CHECK, BACKTRACK, PLAN, ENUMERATE, SETUP, SUMMARIZE, OTHER — and a motif-extraction step characterizes how these primitives compose into recurring sequences (motifs). The framework reveals a depth-recovery tradeoff: RL extends *depth* (longer COMPUTE–CHECK motif chains) but suppresses *recovery* (HYPOTHESIZE and BACKTRACK drop by 70–80%).

This depth-recovery tradeoff we identified motivates a modification to the RL reward: a perplexity-based novelty bonus that rewards rollouts unlikely under a frozen reference model, designed to preserve the recovery primitives vanilla RL suppresses. The bonus restores BACKTRACK to its SFT level and partially restores HYPOTHESIZE, while retaining the depth gains from the COMPUTE–VERIFY chains, adding a further 7pp `pass@32` over a matched vanilla baseline. On problems that only the novelty-trained model solves, recov-

[1]Fin AI Research [2]Salk Institute for Biological Studies. Correspondence to: Mayug Maniparambil <mayugmaniparambil@gmail.com>.

The 3rd *AI for Math Workshop* at the 43rd International Conference on Machine Learning (ICML), Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

ery primitives appear at substantially higher rates, suggesting the bonus works by keeping the model flexible enough to change course when direct execution fails.

Our conclusions — that RL on puzzles transfers, that it does so via primitive-level changes, and that a novelty bonus amplifies the effect — rely on a setup that cleanly isolates RL's contribution. Recent work on puzzle-to-math transfer has not had this property. **Enigmata** (Chen et al., 2025a) trains on a 36-puzzle suite at scale, but includes math problems in both SFT and RL, making it difficult to isolate whether transfer performance originates from in-domain math exposure. **Logic-RL** (Xie et al., 2025) and **LogicPuzzleRL** (Wong et al., 2025) train on puzzles only but evaluate primarily at `pass@1` on benchmarks no harder than AIME24. At `pass@1`, it is impossible to distinguish whether RL expands the set of problems the model can solve at all — capability ceiling expansion — or merely sharpens a solution path it could already reach under sufficient sampling. Whether puzzle-only RL can raise the hard-math capability ceiling therefore remains undemonstrated in the prior literature.

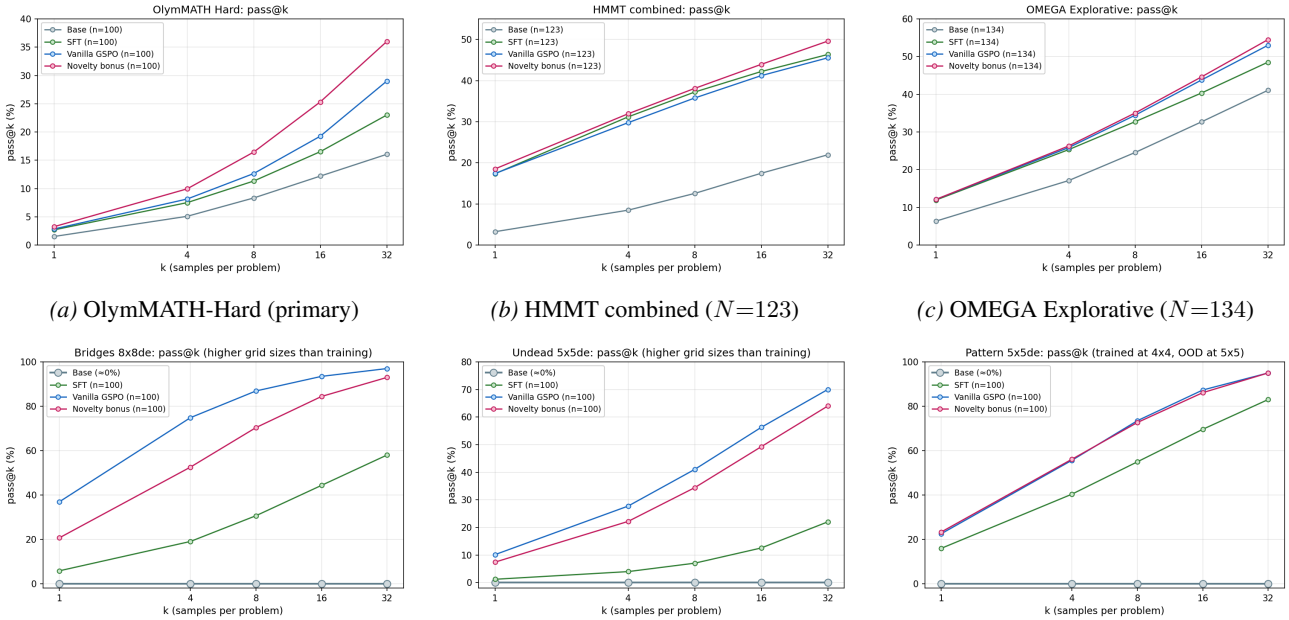
Contributions : (1) We show that RL on a non-mathematical domain (constraint-satisfaction puzzles) transfers to hard mathematics benchmarks, isolating RL's contribution from SFT, (2) We introduce a primitive- and motif-level analysis framework for chain-of-thought reasoning, and use it to identify a depth–recovery tradeoff induced by vanilla RL, (3) We propose a perplexity-based novelty bonus that resolves this tradeoff and yields a further 7pp `pass@32`.

2. Related Work

RL for LLM reasoning and synthetic puzzle tasks. Reinforcement learning from verifiable rewards (RLVR), popularised by DeepSeek-R1 (Guo et al., 2025) and developed through objectives such as GRPO (Shao et al., 2024), GSPO (Zheng et al., 2025), and DAPO (Yu et al., 2025), has become a standard recipe for post-training reasoning models. Synthetic puzzles with deterministic verifiers are a natural testbed for this recipe, and recent works such as Enigmata (Chen et al., 2025a), Logic-RL (Xie et al., 2025), and LogicPuzzleRL (Wong et al., 2025) report transfer from puzzle training to mathematical reasoning. However, existing evidence either mixes math data into the training recipe or evaluates puzzle-only training primarily at `pass@1` on less difficult math benchmarks, leaving open whether non-math RL can expand the mathematical reasoning ceiling, measured by `pass@k` at large k. This question connects to a broader debate about RL post-training transfer: some work reports broad cross-domain gains from math RL (Cheng et al.), while other studies find that single-domain RL mainly sharpens in-domain behaviour (Huan et al., 2025; Hu et al., 2026a; Yue et al., 2025). We study this in a clean isolation: our pipeline uses no math data in SFT or RL and evaluates `pass@k` on hard out-of-domain math. Closely related, Yuan et al. (2025) show that RL can teach models to compose previously learned atomic skills into novel combinations that transfer across tasks. Our observation that vanilla RL chains SFT-introduced primitives into longer compute–verify motifs is consistent with this view, but our contribution is distinct: we isolate cross-domain transfer with no mathematics in post-training, and our central findings concern the *suppression* of exploratory primitives under vanilla RL and their restoration via a novelty bonus, which that line does not address.

Exploration and diversity in RL fine-tuning. Recent work has argued that RLVR can improve `pass@1` while reducing `pass@k` diversity, motivating method families that preserve exploration during RL fine-tuning (Yue et al., 2025; Cui et al., 2025; Yu, 2025; Chen et al., 2025b). One family modifies token-level signals via covariance-based clipping, entropy-term additions, or low-probability regularisation (Cui et al., 2025; Cheng et al., 2025; Huang et al., 2025). A second family reweights whole trajectories by rarity, using current-policy ranks, strategy clusters, or outcome-level novelty (He et al., 2025; Hu et al., 2026b; Song et al., 2025). Our novelty bonus belongs to the trajectory-level family but differs from prior methods in its anchor: novelty is scored under a frozen reference model rather than the current policy, so the diversity target does not drift as training progresses. The bonus applies only to verifier-correct rollouts and uses top-k token NLL rather than LLM-judge clustering, rewarding alternative successful reasoning paths without per-trace judge calls at training time. Unlike work that optimises `pass@k` directly (Walder & Karkhanis, 2025), we retain the standard verifiable-reward objective and use novelty as an auxiliary exploration-preserving bonus.

Mechanism analyses of reasoning. A growing literature analyzes reasoning mechanisms by labeling behaviors in chain-of-thought traces, including verification, backtracking, subgoal setting, and related cognitive operations (Gandhi et al., 2025; Kim et al., 2025); closest to our taxonomy, Li et al. (2025) segment reasoning into Schoenfeld-style functional episodes (e.g. analyze, explore, implement, verify). These studies show that RL reshapes the distribution of reasoning primitives, amplifying some and suppressing others (Gandhi et al., 2025; Bai et al., 2025). Our analysis differs in scale and method: we distill judge annotations into a lightweight span classifier that can label 25k-token traces across tens of thousands of rollouts and ten checkpoints, avoiding both per-trace judge calls and injected behavior markers. Unlike prior analyses, we use the resulting motif distributions to study out-of-domain transfer rather than



(a) OlymMATH-Hard (primary) *(b)* HMMT combined (N=123) *(c)* OMEGA Explorative (N=134)

(d) Bridges 8×8 *(e)* Undead 5×5 *(f)* Pattern 5×5

Figure 1. `pass@k` across the recipe. Top: held-out math benchmarks; bottom: larger held-out puzzle grids than those used in training. Puzzle-only post-training raises `pass@k` on both math and higher-grid-size puzzles; additional math benchmarks are in Appendix F.

tracking only per-trace primitive counts.

3. Post-Training Pipeline and Results

Setup and evaluation. We use OLMo3-7B-Instruct-SFT (Team Olmo et al., 2025) as our base model: it is fully open, has undergone no prior RL fine-tuning, and retains upstream mathematical knowledge from pretraining and instruction tuning. This last property is critical as any post-training gain reflects improved search and verification over existing knowledge, not the injection of new mathematical content. Our post-training pipeline uses only constraint-satisfaction puzzles from the Simon Tatham collection (Tatham, 2024; Estermann et al., 2024) — Bridges, Pattern, Undead, and Galaxies — with no mathematics problems in either the SFT or RL stages. We evaluate on three hard out-of-domain mathematics benchmarks: OlymMATH-Hard (Sun et al., 2025a), HMMT (Balunović et al., 2025), and OMEGA (Sun et al., 2025b), all of which remain far from saturation for a 7B model and therefore probe the capability ceiling rather than benchmark familiarity. Our primary metric is `pass@32`, since a `pass@1` improvement can reflect either capability ceiling expansion, where new problems become solvable, or sharpening of an already-reachable path, and the two are indistinguishable at the single-sample level. Large-k pass rates isolate the former, and all evaluations use 32 rollouts per problem at temperature T=0.6.

Training Recipe. The training pipeline has two stages. First, because the base model has near-zero puzzle competence, direct verifier RL would provide little reward variance, so we perform supervised fine-tuning on rejection-sampled reasoning traces distilled from Deep-Seek Reasoner (DSR), keeping only traces that verify as correct under the puzzle scorers. Second, from this SFT checkpoint we train two multi-puzzle variants of the RL algorithm Group Sequence Policy Optimization (GSPO) (Zheng et al., 2025). Both variants use the same verifier reward, which combines an exact reward for fully correct boards with a partial completeness reward. The partial completeness reward is proportional to how much of the final board state matches the solution. The vanilla GSPO condition uses only this verifier reward, while the novelty condition additionally rewards correct rollouts whose reasoning traces are assigned low probability under the frozen SFT reference. We use GSPO rather than token-level Group Relative Policy Optimization (GRPO) because our target responses are 25–28k tokens and GSPO trained reliably in long-context pilots where GRPO was unstable. Full hyperparameters and reward details are in Appendix K.

Results. Puzzle SFT alone raises OlymMATH-Hard `pass@32` from 16.0% at the base to 23.0%. Both RL variants, trained from this SFT checkpoint, improve further: vanilla GSPO reaches 29.0% (+6pp over SFT), while the novelty variant reaches **36.0**% (+13pp over SFT and +7pp over the matched vanilla GSPO baseline), as shown in Fig-

ure 1 and Table 3.[1] Notably, the novelty gain grows with k—modest at pass@1 but +7pp at pass@32—showing that it increases the capability ceiling rather than sharpening a dominant path. The same pattern holds across other tested benchmarks: on OMEGA Explorative (Sun et al., 2025b), a synthetic benchmark of structured mathematical problems at increased complexity, the novelty variant reaches 54.5% against 48.5% for SFT; on HMMT (Balunović et al., 2025) (123 competition problems), it reaches 49.6% against 46.3%. Per-problem solve rates further support this reading (Appendix Figure 7): Base→Novelty newly solves 29 of 100 OlymMATH-Hard problems while regressing on only 5, confirming the recipe lifts the ceiling broadly rather than sharpening a few already-reachable problems. Finally, on larger-grid puzzle evaluations, both RL variants raise pass@k over SFT across the full sampling budget on Bridges 8×8 and Undead 5×5, suggesting the learned strategies extrapolate across grid scale.

The remainder of the paper asks why puzzle-only training transfers to hard mathematics: what reasoning behaviors does SFT introduce, what does vanilla RL amplify or suppress, and which changes predict the sampled ceiling gains observed above? Section 4 introduces the primitive-level analysis, Sections 4.1–4.3 apply it to the recipe stages, and Sections 5–5.1 use the resulting diagnosis to derive and validate the novelty bonus.

4. Analysis Framework: Reasoning Primitives

Aggregate metrics such as response length and token entropy reveal that training changes the policy (Shypula et al., 2025; Deshpande et al., 2025), but not *what* reasoning behaviours change. We therefore analyse chain-of-thought traces as sequences of reasoning primitives: short, semantically coherent spans that play canonical roles such as planning, computing, checking, hypothesising, or backtracking. Because traces can reach ~ 25k tokens, we split each trace into spans at paragraph boundaries and discourse markers (e.g., *Case 1*, *Wait*, *Alternatively*, *Going back*), yielding contiguous spans of 80–250 tokens (full algorithm in Appendix L). Each span is then classified with its preceding context, producing one primitive sequence per rollout.

Table 1 gives the nine-class vocabulary. We developed the labels using a jury of LLM judges (DeepSeek-V3.2 and DeepSeek-Chat (Liu et al., 2025)), iterating until inter-judge agreement reached 86% on a held-out span set. Because judge-based labelling is impractical at scale, we distil the judges into a nine-way span classifier that reaches 0.80 macro-F1 against judge consensus.

[1]Each RL arm is a single training run (one seed), and the reported RL pass@32 values are read at each arm's best test-set checkpoint (vanilla step 20, novelty step 15). Checkpoint selection is symmetric across arms, so the marginal novelty-over-vanilla gap should be read as a single-seed, best-step comparison.

Table 1. Primitive vocabulary used by the span classifier.

Primitive	Meaning	Primitive	Meaning
PLAN	High-level approach.	SETUP	Notation or auxiliaries.
ENUMERATE	Cases or candidates.	HYPOTHESIZE	Tentative claim to test.
COMPUTE	Calculation or manipulation.	CHECK	Verify against constraints.
BACKTRACK	Reject and return.	SUMMARIZE	Consolidate partial results.
OTHER	Filler or formatting.		

Finally, we convert each primitive sequence into motifs — k-grams of primitives for $k = 2, \ldots, 15$ — and for each checkpoint track primitive frequencies, motif frequencies, and their cross-checkpoint correlation with pass@k (Appendix L). Tracking these statistics across checkpoints lets us identify which reasoning patterns each training stage amplifies or suppresses, and which correlate with hard-math ceiling gains.

4.1. Puzzle SFT Induces a Reasoning Vocabulary

Before any RL, we ask what puzzle SFT alone contributes to reasoning behaviour. Applying our primitive classifier (Section 4), we compare base, DSR-teacher, and post-SFT rollouts on held-out puzzles, and base versus post-SFT rollouts on OlymMATH-Hard. If puzzle-only SFT transfers, the math rollouts should acquire the same primitive vocabulary even though no mathematics appears in the SFT corpus. Figure 2 tests this at both the primitive and trigram-motif levels.

Puzzle SFT induces the vocabulary used later. On held-out puzzles, post-SFT rollouts broadly inherit the DSR teacher's primitives and motif composition, showing that puzzle SFT induces the teacher's reasoning style in addition to improving test accuracy. The recovery primitives HYPOTHESIZE and BACKTRACK are amplified relative to the base model, while the main exploitation primitives remain close to the teacher distribution (Appendix G).

The transfer tests address whether this vocabulary appears out-of-domain. On OlymMATH-Hard, the base model has little structured primitive vocabulary: only 2.8 primitive spans per trace, with essentially no HYPOTHESIZE or BACKTRACK. After puzzle SFT, aggregate primitive count rises to 19.6, CHECK grows from 0.1 to 4.0, and HYPOTHESIZE/BACKTRACK appear at 2.0 and 0.4 episodes per trace. Since the SFT corpus contains only puzzle traces, these changes indicate that puzzle SFT induces a reusable reasoning vocabulary. At the motif level, recovery, exploitation, and verification trigrams are likewise introduced by SFT (definitions in Appendix L), establishing the reasoning vo-

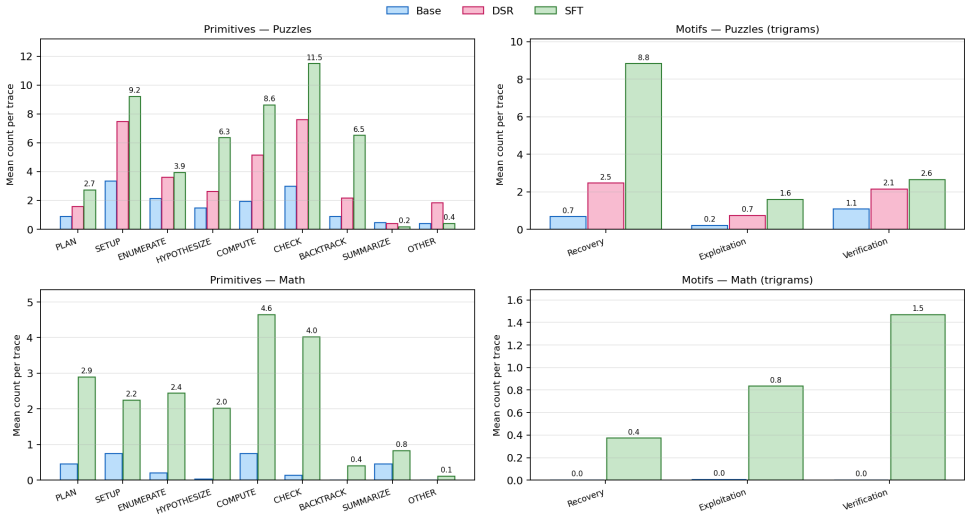


Figure 2. Per-trace primitive counts (left) and trigram-motif counts (right) after puzzle SFT. The top row compares base, DSR-teacher, and post-SFT rollouts on held-out puzzles; the bottom row compares base and post-SFT rollouts on OlymMATH-Hard. DSR is absent from the math row because it was not evaluated on math. Motif definitions are in Appendix L.

cabulary that the following sections examine under RL.

Having established that puzzle SFT makes these primitives available, we next ask how vanilla RL reallocates them.

4.2. What Vanilla RL Changes: Deeper Compute–Verify Chains

What does vanilla RL change relative to puzzle SFT? Our answer is not that it discovers a new reasoning vocabulary. The SFT model already has the relevant primitives; vanilla GSPO instead changes how strongly the model commits to one part of that vocabulary. On OlymMATH-Hard, the dominant shift is toward deeper exploitation: longer contiguous runs of COMPUTE, CHECK, and SETUP episodes that carry an approach forward once the model is on track. The same depth allocation also appears on harder puzzles, paralleling the math-side shift.

From primitives to execution chains. At the primitive level, the three exploitation primitives all rise on OlymMATH-Hard (Figure 3(a)): COMPUTE nearly doubles (4.6 $\rightarrow$ 8.1 episodes per trace), CHECK grows from 4.0 to 5.6, and SETUP from 2.2 to 3.5. The same shift appears in concrete motifs (Figure 3(b)). Short compute–verify patterns increase, but the larger effect is on longer alternations such as CHECK$\rightarrow$COMPUTE$\rightarrow$CHECK$\rightarrow$COMPUTE$\rightarrow$CHECK, which rises 3.4$\times$ from SFT to vanilla GSPO. This is the first sign that vanilla RL is not merely producing more local verification; it is assembling longer execution chains from the same primitives SFT introduced.

A parallel depth shift on hard puzzles. Frequency alone does not tell us whether these primitives are chained together, so we next measure depth directly. We define chain depth as the longest contiguous run of exploitation episodes in a trace before any other primitive intervenes (Appendix M). On OlymMATH-Hard, mean chain depth doubles from SFT to vanilla GSPO (3.2 $\rightarrow$ 6.6), and the 90-th percentile rises from 7 to 14 (Figure 4(a)). The same depth allocation appears on puzzles: holding the checkpoint fixed, chain depth increases on harder puzzle types and larger grids, and vanilla GSPO allocates depth more aggressively than SFT (Appendix I). To check that this is not driven by a few exceptional traces, we also measure the mean length of all exploit runs per trace; that measure rises from 1.74 after SFT to 3.06 after vanilla GSPO (Figure 4(b)). Thus vanilla RL learns extended compute–verify execution on harder puzzle instances.

On OlymMATH-Hard, this depth shift tracks the vanilla-RL gain: `pass@32` rises from 23.0% at SFT to 29.0%, a +6pp gain that accompanies the depth doubling above. This gain represents a ceiling extension — in contrast to the convergence pattern of Yue et al. (2025), puzzle RL adds genuinely new solvable problems rather than rearranging the SFT distribution. Along the Base$\rightarrow$SFT$\rightarrow$vanilla-GSPO trajectory, mean exploit-run length and `pass@32` rise together (Figure 4(b)). Paired inspection of SFT and vanilla-GSPO rollouts on problems solved by both checkpoints gives the same interpretation: vanilla GSPO usually improves *execution* of an approach rather than changing the *set* of approaches attempted (Appendix R, Part A). The novelty-bonus variant later breaks this depth trend, gaining another +7pp `pass@32` with shorter exploit runs; the next

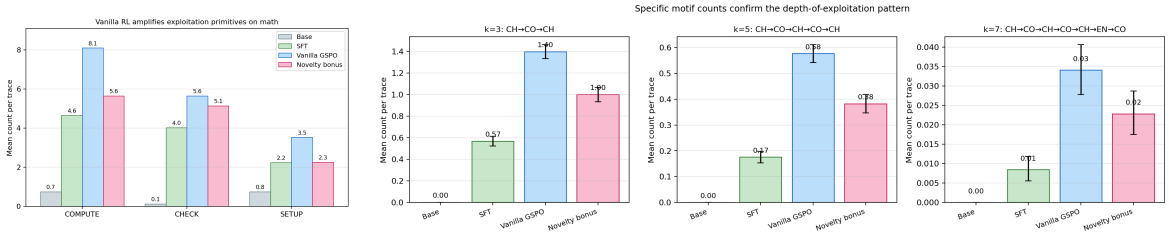


Figure 3. The SFT→vanilla-GSPO transition amplifies exploitation on OlymMATH-Hard. **(a)** COMPUTE, CHECK, and SETUP rise under vanilla GSPO, with a modest reduction under the novelty bonus. **(b)** The same shift appears in k-gram motifs, especially longer compute–verify chains; error bars are 95% CI.

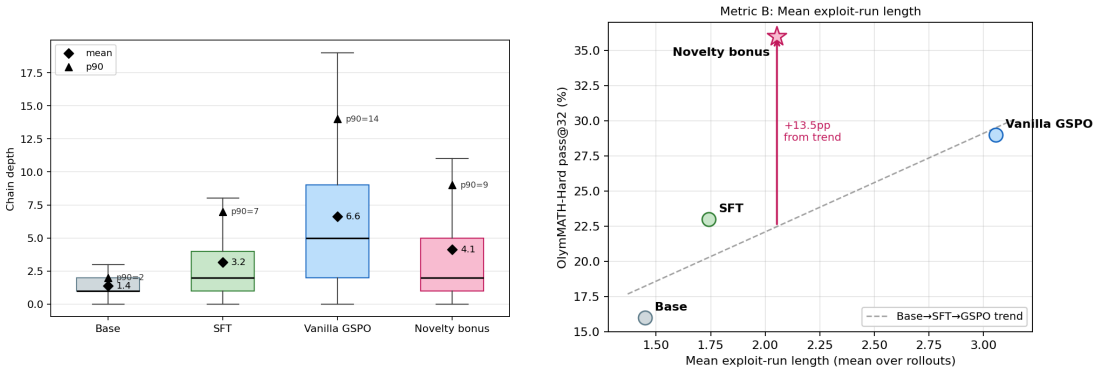


Figure 4. Depth statistics on OlymMATH-Hard. **(a)** Vanilla GSPO roughly doubles mean chain depth and raises the upper tail relative to SFT. **(b)** Mean exploit-run length increases along the Base→SFT→vanilla-GSPO trajectory, while the novelty-bonus variant achieves higher `pass@32` with shorter exploit runs.

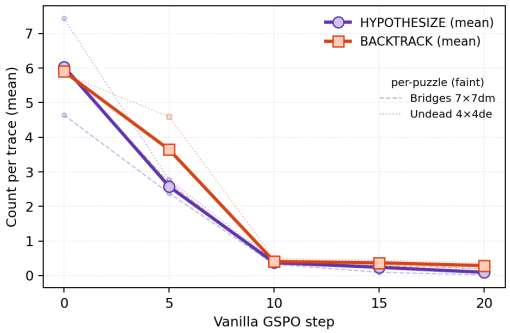
section shows what vanilla RL suppresses.

4.3. What Vanilla RL Suppresses

The depth gain in Section 4.2 comes with a cost. As vanilla GSPO commits more strongly to compute–verify execution, it suppresses the exploratory primitives that puzzle SFT introduced. On OlymMATH-Hard, HYPOTHESIZE falls from 2.0 to 0.6 episodes per trace (-70%), and BACKTRACK from 0.41 to 0.08 (-80%). On the puzzle training distribution itself the suppression is sharper and visibly developmental: across 20 vanilla-GSPO steps, both primitives fall monotonically by 95–98% while puzzle solve rates rise substantially (Figure 5(a)). The novelty bonus restores both primitives toward their SFT levels on math (Figure 5(b)). Vanilla RL therefore does not merely deepen exploitation; it also prunes the recovery vocabulary that would let the model recover when a direct line of reasoning fails.

Why does vanilla RL suppress its own exploratory vocabulary? A simple explanation is that, on the puzzle training distribution, vanilla RL makes the model better at recognizing and executing the right approach. Once the policy can usually identify a productive route, the highest-reward behavior is not to search over alternatives; it is to carry that route farther through setup, computation, and checking — the same pressure that produces the depth increase of Section 4.2. In that regime, HYPOTHESIZE and BACKTRACK introduce detours without often changing the outcome, so continued on-policy training can make them less likely. Figure 5(a) is consistent with this interpretation: as puzzle accuracy rises, the policy gives up the recovery moves that would help when its first approach is wrong.

But hard math is not the training distribution. That tradeoff is useful only when the first approach is usually right. Hard math often violates this assumption: a promising route may lead to a dead end, and solving requires trying another representation, case split, or construction. A model whose HYPOTHESIZE and BACKTRACK primitives have been suppressed can still execute a chosen approach deeply, but it is less able to recover when that approach fails. The novelty GSPO variant, also trained from the SFT checkpoint, preserves both primitives toward their SFT levels on math and raises OlymMATH-Hard `pass@32` to 36.0%, compared with 29.0% for vanilla GSPO and 23.0% for SFT (Table 3); Section 5.1 shows that the gain is concentrated on problems whose solved traces use exactly this recovered



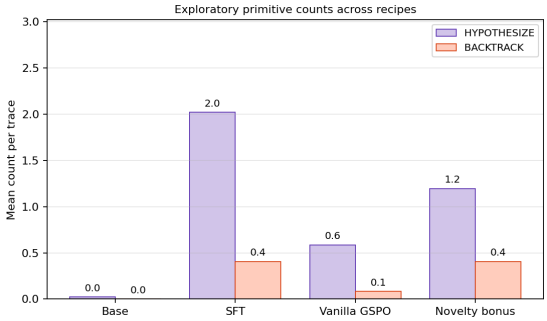


Figure 5. Vanilla RL suppresses HYPOTHESIZE and BACKTRACK. **(a)** On the puzzle training distribution, both primitives fall across vanilla-GSPO training as puzzle pass@k rises. **(b)** The same suppression appears on OlymMATH-Hard; the novelty bonus preserves both primitives closer to their SFT levels.

structure.

Section 5 introduces a targeted intervention designed to preserve the recovery primitives this section showed vanilla RL suppresses.

5. A Frozen-Reference Novelty Bonus Preserves Recovery

Sections 4.2 and 4.3 give us a design target: preserve the recovery primitives that vanilla RL suppresses without giving up the compute–verify depth it learns. We do this by adding a reward bonus for correct rollouts whose reasoning is unusually surprising under the frozen SFT reference model. The bonus is applied only to verifier-correct rollouts and normalised within the correct rollouts on the same prompt: it rewards alternative successful reasoning paths, not arbitrary low-probability text.

Bonus definition. For each rollout r on prompt p, we evaluate the generated tokens under the frozen SFT reference π_{SFT}, with token loss $\text{NLL}_t(r) = -\log \pi_{\text{SFT}}(r_t \mid r_{<t}, p)$. We define the rollout novelty score as the mean of the top-k largest token NLLs:

$$s(r) = \frac{1}{k} \sum_{t \in \text{top}_k(\text{NLL}(r))} \text{NLL}_t(r), \qquad k = 100. \tag{1}$$

For correct rollouts within each prompt group, we z-score and clip the novelty scores before adding them to the verifier reward:

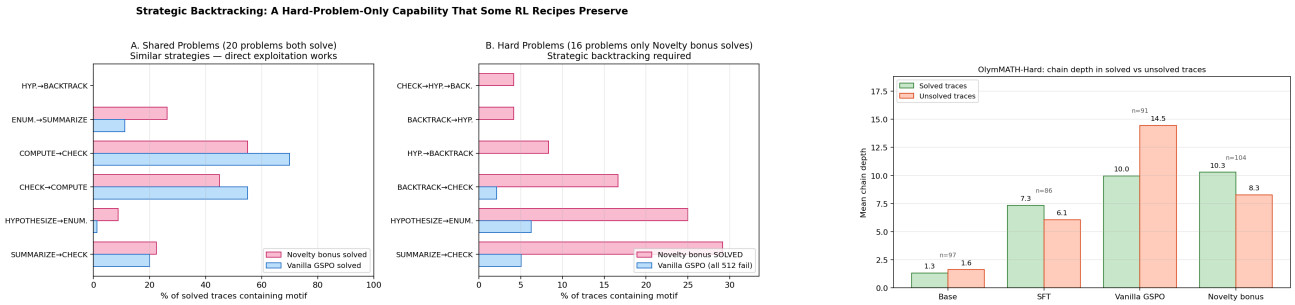
$$z_p(r) = \text{clip}\left(\frac{s(r) - \mu_p}{\sigma_p},\ -2,\ 2\right), \tag{2}$$

$$R_{\text{novelty}}(r) = R_{\text{base}}(r) + \alpha z_p(r)\mathbb{1}[r \text{ correct}]. \tag{3}$$

where μ_p and σ_p are computed over correct rollouts for prompt p. We set $\alpha = 0.1$. Thus incorrect rollouts receive no novelty credit, and correct rollouts receive more credit when their reasoning is more novel relative to the SFT reference than other correct rollouts on the same prompt. Pseudocode is in Appendix O.

Why frozen-reference top-k NLL? The frozen reference keeps the novelty target stable. If novelty were ranked under the current policy, as in unlikeliness-style rewards (He et al., 2025), the target would drift toward whatever the model is already doing well, weakening the diversity signal as training progresses. The top-k statistic makes the signal local enough to matter at our trace lengths: with $\sim$25k-token rollouts, mean NLL has near-zero within-group variance because most tokens are high-probability formatting or structural choices. Using only the $k = 100$ highest-NLL tokens isolates rare decision points and amplifies the within-group signal by 10–15$\times$ (Appendix P).

Effect. The novelty-bonus variant starts from the same SFT model as vanilla GSPO and uses the same data, algorithm, training budget, and number of optimisation steps. It raises OlymMATH-Hard `pass@32` to **36.0**%, compared with 29.0% for vanilla GSPO and 23.0% for the shared SFT starting point (Table 3; Figure 1a). Mechanistically, it restores the recovery primitives that vanilla GSPO suppressed: BACKTRACK returns to its SFT level and HYPOTHESIZE recovers to about 60% of its SFT rate (Figure 5b). This restoration comes with less exploitation depth: chain depth, exploit-run length, and the exploitation motifs amplified by vanilla GSPO all reduce modestly (Figures 4(b) and 3(b)). The bonus therefore does not improve by adding still more depth; it trades some direct compute–verify execution for recovered ability to try and abandon alternatives. The same trade appears in domain: compared at matched best-puzzle checkpoints, the novelty bonus lowers in-domain accuracy on some puzzles (most notably Bridges, -29.5pp) while improving others (Galaxies, Pattern; Table 2), reshaping in-domain skill rather than uniformly degrading it.



(a) Recovery motifs by problem subset

(b) Depth split by trace success

Figure 6. Diagnostic evidence for the novelty gain on OlymMATH-Hard. **(a)** On shared problems, vanilla GSPO and novelty rollouts have similar motif distributions; on unique-to-novelty problems, recovery motifs remain available only under the novelty bonus. **(b)** Vanilla GSPO spends more depth on failures, while the novelty-bonus variant reverses the sign; significance tests are in Table 9.

We also screened diversity-promoting alternatives in smaller pilots, including a loss-level entropy bonus and entropy-advantage variants inspired by Cheng et al. (2025). These pilot experiments were used for method selection, not as a definitive full-scale comparison: entropy matched the novelty bonus at mini scale but was less stable, while entropy-advantage variants collapsed below baseline. We therefore scaled the frozen-reference novelty bonus for the GSPO comparison; the pilot summary and its limitations are in Appendix N. The next section examines whether the restored recovery primitives account for the performance gain.

5.1. Where the Novelty Gain Comes From: Recovery on Unique-to-Novelty Problems

Section 5 left a specific question: does the novelty bonus restore HYPOTHESIZE/BACKTRACK everywhere, or does it deploy recovery selectively on the problems where direct compute–verify execution fails? We test this on the 100 OlymMATH-Hard problems by comparing two diagnostic subsets: problems solved by both vanilla GSPO and the novelty-bonus variant ("shared"), and problems solved only by the novelty-bonus variant ("unique-to-novelty"). The full accounting also includes vanilla-only and unsolved problems (Appendix Q), but the contrast between shared and unique-to-novelty problems is the cleanest test of selectivity: shared problems show whether the novelty-bonus variant behaves like vanilla GSPO when direct exploitation is enough, while unique-to-novelty problems show whether recovery appears where vanilla GSPO fails.

Recovery motifs are selective, not uniform. The aggregate motif analysis shows where the novelty bonus changes behavior (Figure 6(a)). On the 20 shared problems, solved traces from vanilla GSPO and the novelty-bonus variant look nearly identical: both rely on direct COMPUTE→CHECK exploitation, and recovery motifs remain rare. On the 16 unique-to-novelty problems, the pattern changes. The recovery motifs are absent from vanilla GSPO's 512 attempts but remain available in the novelty-bonus variant ($11/512$ rollouts). Within the novelty-bonus run, recovery motifs are also concentrated in successful traces: $3/24$ solved traces contain at least one diagnostic recovery motif versus $8/488$ failed traces (12.5% vs 1.6%; Fisher exact $p = 0.012$). The counts are small, so we read this as a directional enrichment rather than a precise effect size, but it indicates the gain is not explained by recovery being used uniformly across novelty rollouts. Instead, the bonus preserves a recovery move that vanilla GSPO has lost, and that move appears disproportionately in successful unique-to-novelty traces. This is the selectivity pattern Section 4.3 predicted: recovery is unnecessary on shared problems but most useful on problems where vanilla GSPO's direct execution fails. Appendix R, Part B, gives two full trace pairs showing the same pivot in text.

Novelty aligns depth with successful trajectories. Section 4.2 left a puzzle: the novelty-bonus variant achieves higher `pass@32` than vanilla GSPO despite shorter average exploit runs. Splitting traces by whether they solve the problem clarifies this pattern. Vanilla GSPO is the only RL run whose unsolved traces are deeper than its solved ones; the novelty-bonus variant reverses the sign, with solved traces deeper than unsolved (Figure 6(b)). This suggests that vanilla GSPO often spends depth on unsuccessful continuations, whereas the novelty bonus makes depth more coupled to trajectories that succeed.

6. Conclusion and Limitations

We post-trained a 7B model with no mathematics in any post-training stage and found that puzzle-only SFT and RL can raise the sampled ceiling on hard mathematics. The primitive analysis suggests a more specific lesson than "more reasoning helps." Puzzle SFT makes a reusable vocabulary of planning, checking, hypothesizing, and backtracking available out of domain; vanilla GSPO then sharpens execution by lengthening compute–verify chains, but at the

cost of suppressing recovery moves. The frozen-reference novelty bonus improves transfer by preserving some of that recovery vocabulary while retaining enough exploitation depth. More broadly, cross-domain RL transfer appears to depend not only on how far a model can pursue an approach, but also on whether training preserves the ability to abandon that approach when it fails. Taken together, our contribution is an empirical test of whether RL-trained LLMs can turn a shared problem-solving vocabulary into measurable out-of-domain math gains.

Limitations. Our results are limited to one base model (OLMo3-7B-Instruct-SFT) and one RL family (GSPO); the novelty-bonus hyperparameters were selected from pilot runs rather than exhaustive tuning; and the primitive analysis is a diagnostic lens, not a per-problem causal proof. Extended discussion is in Appendix C.

Acknowledgments

We thank Fin AI Research for supporting and funding this work.

References

Bai, H., Sun, Y., Hu, W., Qiu, S., Huan, M. Z., Song, P., Nowak, R., and Song, D. X. How and why llms generalize: A fine-grained analysis of llm reasoning from cognitive behaviors to low-level patterns. *ArXiv*, abs/2512.24063, 2025. URL https://api.semanticscholar.org/CorpusID:284350292.

Balunović, M., Dekoninck, J., Petrov, I., Jovanović, N., and Vechev, M. Matharena: Evaluating llms on uncontaminated math competitions. *arXiv preprint arXiv:2505.23281*, 2025. doi: 10.48550/arXiv.2505.23281. URL https://arxiv.org/abs/2505.23281.

Chen, J., He, Q., Yuan, S., Chen, A., Cai, Z., Dai, W., Yu, H., Chen, J., Li, X., Yu, Q., Zhou, H., and Wang, M. Enigmata: Scaling logical reasoning in large language models with synthetic verifiable puzzles. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=fmnxunacr4.

Chen, Z., Qin, X., Wu, Y., Ling, Y., Ye, Q., Zhao, W. X., and Shi, G. Pass@ k training for adaptively balancing exploration and exploitation of large reasoning models. *arXiv preprint arXiv:2508.10751*, 2025b.

Cheng, D., Huang, S., Zhu, X., Dai, B., Zhao, W. X., Zhang, Z., and Wei, F. Reasoning with exploration: An entropy perspective. In *AAAI Conference on Artificial Intelligence*, 2025. URL https://api.semanticscholar.org/CorpusID:279410424.

Cheng, Z., Hao, S., Liu, T., Zhou, F., Xie, Y., Yao, F., Bian, Y., Dey, N., Zhuang, Y., Zha, Y., et al. Revisiting reinforcement learning for llm reasoning from a cross-domain perspective. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Cui, G., Zhang, Y., Chen, J., Yuan, L., Wang, Z., Zuo, Y., Li, H., Fan, Y., Chen, H., Chen, W., Liu, Z., Peng, H., Bai, L., Ouyang, W., Cheng, Y., Zhou, B., and Ding, N. The entropy mechanism of reinforcement learning for reasoning language models. *ArXiv*, abs/2505.22617, 2025. URL https://api.semanticscholar.org/CorpusID:278959427.

Deshpande, V., Ghose, D., Patterson, J. D., Beaty, R., and Rumshisky, A. Diverse, not short: A length-controlled data selection strategy for improving response diversity of language models. *arXiv preprint arXiv:2505.16245*, 2025. URL https://arxiv.org/abs/2505.16245.

Estermann, B., Lanzendörfer, L. A., Niedermayr, Y., and Wattenhofer, R. Puzzles: A benchmark for neural algorithmic reasoning. *Advances in Neural Information Processing Systems*, 37:127059–127098, 2024.

Gandhi, K., Chakravarthy, A., Singh, A., Lile, N., and Goodman, N. D. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective STaRs. *arXiv preprint arXiv:2503.01307*, 2025. COLM 2025.

Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., Zhang, R., Ma, S., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

He, A., Fried, D., and Welleck, S. Rewarding the unlikely: Lifting GRPO beyond distribution sharpening. *arXiv preprint arXiv:2506.02355*, 2025. EMNLP 2025.

Hu, C., Zhu, Y., Kellermann, A., Biddulph, C., Waiwitlikhit, S., Benn, J., and Kang, D. Breaking barriers: Do reinforcement post training gains transfer to unseen domains? In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=mvLhN0veUd.

Hu, Z., Wang, Y., He, Y., Wu, J., et al. Rewarding the rare: Uniqueness-aware reinforcement learning for creative problem solving. *arXiv preprint arXiv:2601.08763*, 2026b.

Huan, M. Z., Li, Y., Zheng, T., Xu, X., Kim, S., Du, M., Poovendran, R., Neubig, G., and Yue, X. Does math reasoning improve general llm capabilities? understanding transferability of llm reasoning. *ArXiv*, abs/2507.00432, 2025. URL https://api.semanticscholar.org/CorpusID:280146966.

Huang, G., Xu, T., Wang, M., Yi, Q., Gong, X., Li, S., Xiong, R., Li, K., Jiang, Y., and Zhou, B. Low-probability tokens sustain exploration in reinforcement learning with verifiable reward. *ArXiv*, abs/2510.03222, 2025. URL https://api.semanticscholar.org/CorpusID:281830220.

Kim, J., Goyal, A., Tan, L., Hajishirzi, H., Iyer, S., and Wang, T. Astro: Teaching language models to reason by reflecting and backtracking in-context. *ArXiv*, abs/2507.00417, 2025. URL https://api.semanticscholar.org/CorpusID:280128357.

Li, M., Fan, C., Cheng, Y., Feizi, S., and Zhou, T. Schoenfeld's anatomy of mathematical reasoning by language models. *arXiv preprint arXiv:2512.19995*, 2025.

Liu, A., Mei, A., Lin, B., Xue, B., Wang, B., Xu, B., Wu, B., Zhang, B., Lin, C., Dong, C., et al. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.

Pólya, G. *How to Solve It: A New Aspect of Mathematical Method*. Princeton University Press, Princeton, NJ, 1945.

Schoenfeld, A. H. *Mathematical problem solving*. Elsevier, 2014.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y. K., Wu, Y., and Guo, D. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Shypula, A., Li, S., Zhang, B., Padmakumar, V., Yin, K., and Bastani, O. Evaluating the diversity and quality of llm generated content. *arXiv preprint arXiv:2504.12522*, 2025. URL https://arxiv.org/abs/2504.12522.

Simon, H. A. and Newell, A. Human problem solving: The state of the theory in 1970. *American psychologist*, 26(2): 145, 1971.

Song, Y., Kempe, J., and Munos, R. Outcome-based exploration for LLM reasoning. In *NeurIPS 2025 Workshop: Second Workshop on Aligning Reinforcement Learning Experimentalists and Theorists*, 2025. URL https://openreview.net/forum?id=VORSpYLBJ6.

Sun, H., Min, Y., Chen, Z., Zhao, W. X., and Wen, J.-R. Challenging the boundaries of reasoning: An olympiad-level math benchmark for large language models. *arXiv preprint arXiv:2503.21380*, 2025a. OlymMATH benchmark; Easy and Hard splits, 50 problems each.

Sun, Y., Hu, S., Zhou, G., Zheng, K., Hajishirzi, H., Dziri, N., and Song, D. Omega: Can llms reason outside the box in math? evaluating exploratory, compositional, and transformative generalization. *arXiv preprint arXiv:2506.18880*, 2025b. doi: 10.48550/arXiv.2506.18880. URL https://arxiv.org/abs/2506.18880.

Tatham, S. Simon tatham's portable puzzle collection. https://www.chiark.greenend.org.uk/~sgtatham/puzzles/, 2024.

Team, Q. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Team Olmo, Ettinger, A., Bertsch, A., Kuehl, B., Graham, D., Heineman, D., et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025. Base model: allenai/OLMo-3-7B-Instruct-SFT.

Walder, C. and Karkhanis, D. Pass@ k policy optimization: Solving harder reinforcement learning problems. *arXiv preprint arXiv:2505.15201*, 2025.

Wong, Z. H., Deng, J., He, R., et al. LogicPuzzleRL: Cultivating robust mathematical reasoning in LLMs via reinforcement learning on logic puzzles. *arXiv preprint arXiv:2506.04821*, 2025.

Xie, T., Gao, Z., Ren, Q., Luo, H., Hong, Y., Dai, B., Zhou, J., Qiu, K., Wu, Z., and Luo, C. Logic-RL: Unleashing LLM reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*, 2025.

Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., YuYue, Dai, W., Fan, T., Liu, G., Liu, J., Liu, L., Liu, X., Lin, H., Lin, Z., Ma, B., Sheng, G., Tong, Y., Zhang, C., Zhang, M., Zhang, R., Zhang, W., Zhu, H., Zhu, J., Chen, J., Chen, J., Wang, C., Yu, H., Song, Y., Wei, X., Zhou, H., Liu, J., Ma, W.-Y., Zhang, Y.-Q., Yan, L., Wu, Y., and Wang, M. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=2a36EMSSTp.

Yu, Y. Pass@k metric for RLVR: Diagnostic, not objective. *arXiv preprint arXiv:2511.16231*, 2025.

Yuan, L., Chen, W., Zhang, Y., Cui, G., Wang, H., You, Z., Ding, N., Liu, Z., Sun, M., and Peng, H. From $f(x)$ and $g(x)$ to $f(g(x))$: Llms learn new skills in rl by composing old ones. *arXiv preprint arXiv:2509.25123*, 2025.

Yue, Y., Chen, Z., Lu, R., Zhao, A., Wang, Z., Yue, Y., Song, S., and Huang, G. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025.

Zheng, C., Liu, S., Li, M., Chen, X.-H., Yu, B., Gao, C., Dang, K., Liu, Y., Men, R., Yang, A., Zhou, J., and Lin, J. Group sequence policy optimization. *ArXiv*, abs/2507.18071, 2025. URL https://api.semanticscholar.org/CorpusID:280017753.

A. Broader impact

Our contribution improves out-of-domain reasoning of a small open research model via verifier-only puzzle RL; the standard dual-use considerations for reasoning LLMs apply, and the method does not introduce capability surfaces beyond those already accessible from the base model.

B. Compute resources

Stage 1 SFT runs on $8\times$B200 (180 GB) for $\approx$9 hr. The two Stage 2 RL variants—vanilla GSPO and the novelty-bonus GSPO—are parallel branches from the same SFT checkpoint and each runs on $8\times$B200 for $\approx$70 hr. End-to-end reproduction of the headline OlymMATH-Hard 36.0% pass@32 number (SFT plus the novelty-bonus branch) is therefore approximately 632 GPU-hours; reproducing both Stage 2 branches together is approximately 1,192 GPU-hours. The full research project—preliminary pilot runs, abandoned hyperparameter configurations, and ablations not reported in the paper—required roughly an additional 2–3$\times$ the reproduction budget on top.

C. Limitations and future work

Our evidence comes from one base model, OLMo3-7B-Instruct-SFT, and one RL algorithm family, GSPO. Because full-scale runs with long puzzle rollouts are computationally expensive, we leave validation on other models, scales, and RL objectives to future work. The novelty bonus also has design choices that were not exhaustively tuned at full scale: $\alpha = 0.1$ and top-$k = 100$ were selected from smaller pilot runs. For the same reason, alternative diversity mechanisms were screened in smaller pilots rather than compared at the full training scale. Finally, our primitive analysis is a diagnostic lens, not a causal proof for every solved problem: the classifier was validated on this model family and trace style, and the motif evidence shows where recovery is available and enriched, not that every novelty-only solution uses it. Finally, "no mathematics" refers to the post-training problems and rewards—every SFT and RL instance is a puzzle—but the SFT traces are distilled from Deep-Seek Reasoner, whose own training included mathematics; the imported reasoning primitives may therefore carry a math-shaped origin, so our claim is that puzzle-only *problems* elicit and reshape deployable reasoning, not that the primitives are math-free in lineage.

D. Detailed accuracy tables

This appendix collects the per-benchmark accuracy tables referenced from Section 3. Table 2 reports in-domain puzzle accuracy at `pass@1`; Table 3 reports held-out mathematics at `pass@1` and `pass@32`. Both are restated here for ease of cross-reference; Figure 1 summarises the same data as `pass@`k curves for the body benchmarks.

Table 2. In-domain puzzle accuracy on held-out test sets at training grid sizes, `pass@1` (greedy, 200 problems per grid). We omit `pass@32`: it saturates for the SFT baseline and does not differentiate the recipe stages. Each RL row reports the run's best-puzzle-aggregate step. The novelty bonus's puzzle peak (step 10) precedes its math peak (step 15, the headline math checkpoint in Table 3); we report both for transparency.

	Bridges 7×7	Pattern 4×4	Undead 4×4	Galaxies 3×3
Base	0.0	0.0	0.0	0.0
+ Puzzle SFT (rejection-sampled DSR)	28.0	34.3	22.5	45.0
+ GSPO RL (vanilla; ours)	**79.5**	53.5	**57.0**	65.0
+ GSPO RL with novelty bonus (step 10, best puzzle)	50.0	56.6	47.0	**80.0**
+ GSPO RL with novelty bonus (step 15, best math; ours)	46.0	**66.7**	47.0	67.5

E. Paired significance of the per-problem ceiling flips

The `pass@32` comparisons in Table 3 are paired at the problem level: every checkpoint is evaluated on the same 100 OlymMATH-Hard problems, so each recipe contrast is summarised by how many problems flip at the ceiling in each direction. We test each contrast with an exact two-sided McNemar test on the discordant pairs (unlocked $= 0 \rightarrow \geq 1$ correct in 32 rollouts; lost = the reverse), and report a paired-bootstrap (10^5 resamples) 95% confidence interval on the `pass@32` difference. Base is truncated to its first 32 rollouts for an apples-to-apples `pass@32` comparison. Counts here use the strict ceiling-flip definition and therefore match Figure 7.

Table 3. Capability on held-out mathematics. The recipe trains on constraint-satisfaction puzzles only; OlymMATH and AIME25 never appear in training. We report `pass@1` (accuracy) and `pass@32` (capability ceiling) over 32 rollouts per problem at temperature 0.6. OlymMATH-Hard is the difficulty tier of interest: AIME25 and OlymMATH-Easy saturate quickly with sampling at this scale, so RL gains there are necessarily small. The novelty bonus's largest gain is on the unsaturated benchmark (Hard, +7pp at `pass@32`); on every benchmark its effect on `pass@1` is near zero, the signature of a method that preserves multiple correct reasoning paths rather than sharpening any single one.

	OlymMATH-Hard		AIME25		OlymMATH-Easy	
	@1	@32	@1	@32	@1	@32
Base (OLMo3-7B-Instruct-SFT)	1.5	16.0	9.4	34.0	3.2	33.9
+ Puzzle SFT (rejection-sampled DSR traces)	2.7	23.0	27.0	73.1	14.2	72.0
+ GSPO RL (vanilla; ours)	2.8	29.0	**29.6**	75.4	**15.9**	**73.0**
+ GSPO RL with novelty bonus (ours)	**3.2**	**36.0**	28.5	**76.7**	15.4	71.0

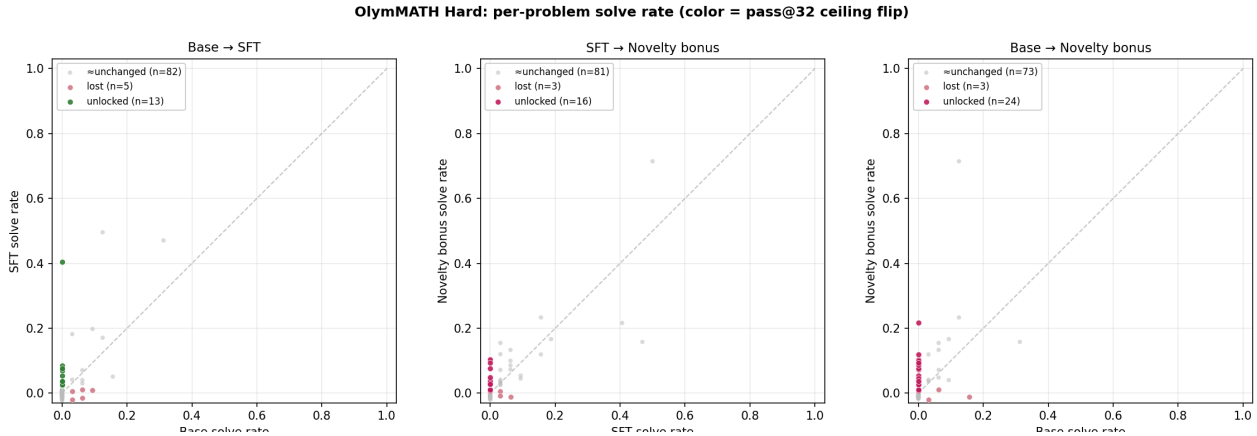


Figure 7. Per-problem solve rates on OlymMATH-Hard (100 problems, 32 rollouts per problem) across the recipe. Each point is one problem; colour marks problems that flip at the `pass@32` ceiling (unsolved ↔ solved, i.e. $0 \rightarrow \geq 1$ correct in 32 rollouts). The ceiling rise is driven by genuine per-problem unlocks: Base→Novelty unlocks 24 problems and loses 3 (8:1), with most unlocks happening past SFT (SFT→Novelty unlocks 16, loses 3). Paired significance for every contrast is in Table 4. The recipe lifts the ceiling on a broad set of problems rather than sharpening sampling on a few already-solved ones.

F. Additional benchmark `pass@`k curves

Figure 1 (body) reports OlymMATH-Hard, HMMT combined, OMEGA Explorative, and the held-out larger-grid puzzle evaluations. We document four additional benchmarks here for completeness: OlymMATH-Easy, AIME25, OMEGA Compositional, and OMEGA Transformative. None earn a body slot — each is either saturated by SFT or has a confound that limits the RL-discriminating zone — but together they verify that the recipe does not regress on benchmarks the body figure does not show.

OlymMATH-Easy and AIME25 saturate quickly with 32 rollouts at this scale, so the gap between recipe stages collapses past pass@8 — both Vanilla GSPO and Novelty are within $\sim$ 1–3pp of each other and within $\sim$ 4pp of SFT on pass@32. They confirm the recipe roughly preserves easier-tier accuracy (differences between RL stages are within the noise floor at saturation) but do not differentiate RL stages. OMEGA Compositional shows a near-tie between SFT and vanilla GSPO at pass@32 (both 56%); novelty lifts to 60%. OMEGA Transformative is the most stress-test-like split but is bimodal at 7B scale: a per-config audit shows only $\sim$24 of 100 problems are in the RL-discriminating zone — the rest are either solved by Base (42 problems) or universally unsolvable at this scale (34 problems, dominated by de Moivre and function-intersection configs where every checkpoint scores zero). Vanilla GSPO regresses by a hair on pass@32 ($43 \rightarrow 42$); novelty recovers and adds +2pp over SFT.

Evaluation cost note. A single pass@32 sweep at `max_gen_toks`=25,000 generates $N \times 32$ rollouts of up to 25,000 tokens each. With seed-diversification required for valid pass@k statistics, the 32 samples per problem must be sampled independently rather than batched as a fixed-prompt expansion; the cost is therefore dominated by autoregressive decoding latency, not forward-pass throughput. On 8× B200 with vLLM tensor-parallel=1, a single 4-checkpoint sweep at N=100

Table 4. Paired significance of OlymMATH-Hard `pass@32` ceiling flips (N=100, single RL seed). The end-to-end and SFT→Novelty contrasts are significant; the single-stage steps—including the marginal vanilla→novelty increment—are within noise at this sample size and seed count, consistent with the single-seed scope of our RL runs.

Contrast	Δ (pp)	unlocked / lost	McNemar p	95% CI (pp)
Base → SFT	$+8$	$13/5$	0.10	$[0, +16]$
SFT → Vanilla GSPO	$+6$	$16/10$	0.33	$[-4, +16]$
Vanilla GSPO → Novelty	$+7$	$16/9$	0.23	$[-3, +17]$
SFT → Novelty	$+13$	$16/3$	0.004	$[+5, +21]$
Base → Novelty	$\mathbf{+21}$	$\mathbf{24/3}$	$\mathbf{5\times10^{-5}}$	$[\mathbf{+12}, \mathbf{+30}]$

takes roughly $11-12$ wall-clock hours; a full 4-checkpoint sweep at N=300 would take $\sim$ 36 hr. We therefore report OMEGA Explorative on its full held-out test split (N=134) and cap the larger OMEGA Compositional and Transformative pools at N=100 problems, sub-sampled with a fixed seed for reproducible apples-to-apples scoring across checkpoints.

G. Per-setting primitive distributions

We apply our classifier of Section 4 to three populations of reasoning traces and report the relative composition of primitive episodes (percent of all classified primitive labels). The three settings are: **DSR puzzles** (reasoning traces from the DSR teacher on the puzzle problems used in our SFT corpus); **SFT puzzles** (post-SFT OLMo3 rollouts on held-out puzzle test sets at training grid sizes); **SFT math** (post-SFT OLMo3 rollouts on OlymMATH-Hard, the same distribution Section 4.1 reports per-trace counts on).

Table 5. Relative primitive composition (% of primitive episodes) across the three settings. For PLAN, COMPUTE, CHECK, and ENUMERATE, the post-SFT distribution on puzzles closely tracks DSR's puzzle distribution, indicating SFT broadly mimics the teacher's reasoning style. Recovery primitives (HYPOTHESIZE, BACKTRACK) are present in DSR at 4.6% and 3.2% and amplified post-SFT to 11.0% and 11.3%. On math (a held-out domain), the SFT distribution shows different relative proportions—more SUMMARIZE, less BACKTRACK—suggesting the model deploys reasoning structure conditional on the input domain.

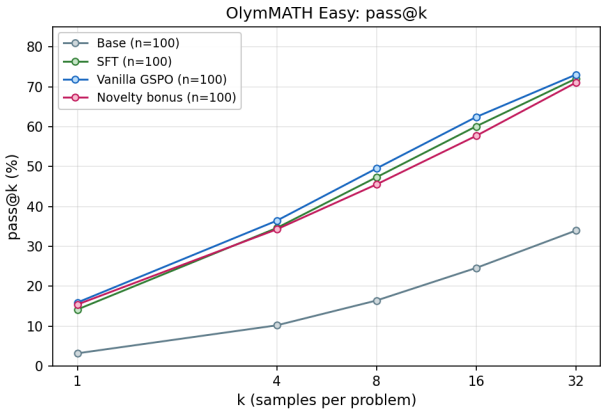
Primitive	DSR puzzles	SFT puzzles	SFT math
PLAN	4.7	5.5	16.7
SETUP	26.8	20.3	18.9
ENUMERATE	9.6	8.4	9.1
HYPOTHESIZE	4.6	11.0	5.4
COMPUTE	17.9	18.3	24.2
CHECK	23.4	23.3	12.7
BACKTRACK	3.2	11.3	0.9
SUMMARIZE	1.6	0.5	11.8
OTHER	8.2	1.5	0.3

H. Mechanism patterns across math benchmarks

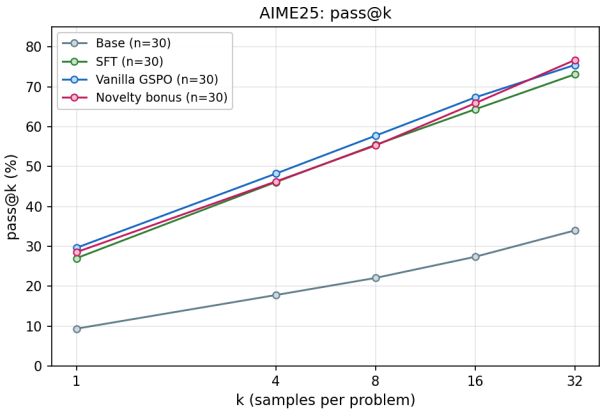
The mechanism analysis in Sections 4.2–4.3 is reported on OlymMATH-Hard. To check that the patterns are not specific to that benchmark, we run the same classifier on rollouts at the four main checkpoints on **OlymMATH-Easy** and **AIME25** and recompute the body's load-bearing metrics with the canonical body definitions (non-overlapping motif counts; chain depth as the longest consecutive run of COMPUTE/CHECK/SETUP). This appendix reports those cross-benchmark numbers; the Hard column reproduces the body figures.

The qualitative finding is that every directional ranking in §4.2–§4.3 holds on Easy and AIME25 with similar magnitudes:

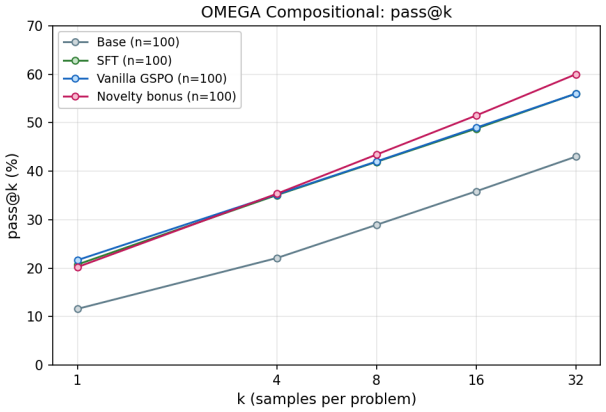
- Chain depth: $\text{Base} < \text{SFT} < \text{Vanilla GSPO}$, novelty sits between SFT and vanilla GSPO.
- HYPOTHESIZE count: suppressed under vanilla GSPO (3–5$\times$ drop from SFT), partially restored under novelty.
- BACKTRACK count: suppressed under vanilla GSPO ($\sim 5\times$), fully restored or exceeded by novelty.
- $k = 3$ and $k = 5$ exploitation motifs (CHECK $\rightarrow$ COMPUTE $\rightarrow \ldots$): rise sharply from SFT to vanilla GSPO, novelty drops back partway.



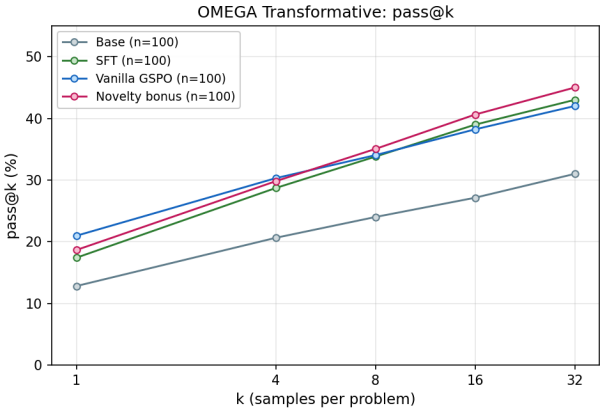
(a) OlymMATH-Easy (N=100)



(b) AIME25 (N=30)



(c) OMEGA Compositional (N=100)



(d) OMEGA Transformative (N=100)

Figure 8. Additional math benchmark `pass@`k curves not shown in the body. **(a)** OlymMATH-Easy: large Base→SFT lift (+38pp pass@32, 34 → 72); RL contribution is small thereafter as the benchmark saturates. **(b)** AIME25: same shape as Easy — Base→SFT carries the bulk of the lift (34 → 73pp), RL stages tied within ∼ 1–3pp at every k. **(c)** OMEGA Compositional (7 configs combining pairs of math domains): Base→SFT +12pp, vanilla GSPO ties SFT, novelty adds +4pp. SFT already covers the relevant compositional skills. **(d)** OMEGA Transformative (7 configs requiring creative reframing): bimodal at 7B — 34 of 100 problems are universally unsolvable (de Moivre, function-intersection), 42 are Base-saturated, only 24 in the RL-discriminating zone. Recipe is non-monotonic (31 → 43 → 42 → 45); vanilla GSPO regresses by 1pp at pass@32, novelty recovers and adds +2pp over SFT.

Where magnitudes shift. AIME25 has slightly higher chain depth at every checkpoint (∼1 unit deeper than Hard) — competition-math problems require longer reasoning chains than the OlymMATH-Hard distribution. Easy has slightly weaker HYP/BTK suppression ratios under vanilla GSPO than Hard, consistent with Easy saturating earlier (SFT `pass@32` = 72%): the SFT distribution there is already close to ceiling, leaving less room for vanilla RL to differentiate. The AIME25 novelty-bonus cell has the smallest sample size (960 traces; 30 docs × 32 rollouts) and is the most sample-size-noisy column. Across all metrics, no benchmark contradicts the direction of any §4.2–§4.3 ranking.

I. Depth shift on puzzles and within-checkpoint scaling

Section 4.2 reports that vanilla RL roughly doubles chain depth on OlymMATH-Hard. We verify here that (i) the same depth shift is present on harder OOD puzzles—so the math depth shift is not a math-specific artefact—and (ii) within-checkpoint, depth scales with task complexity, ruling out a mechanical "bigger-input-longer-output" reading.

The SFT→vanilla GSPO depth shift appears on OOD puzzles too. On *solved* traces at the largest puzzle grids in our test suite (held out from training, which used smaller grids), mean chain depth rises substantially from SFT to vanilla GSPO. On Bridges 8×8, depth goes from 13.6 [13.1, 14.2] at SFT (n_{solved}=185) to 19.4 [19.1, 19.7] at vanilla GSPO

Table 6. Mean chain depth (longest exploit run per trace).

Checkpoint	OlymMATH-Hard	OlymMATH-Easy	AIME25
Base	1.39	1.56	1.53
SFT	3.17	3.52	4.40
Vanilla GSPO	6.63	6.68	7.67
Novelty bonus	4.14	4.57	5.16

Table 7. Mean HYPOTHESIZE (left) and BACKTRACK (right) counts per trace.

	HYPOTHESIZE			BACKTRACK		
Checkpoint	Hard	Easy	AIME25	Hard	Easy	AIME25
Base	0.02	0.02	0.01	0.00	0.00	0.01
SFT	2.02	1.11	0.96	0.41	0.31	0.29
Vanilla GSPO	0.59	0.29	0.19	0.08	0.08	0.05
Novelty bonus	1.20	0.73	0.55	0.41	0.37	0.24

(n_{solved}=1181)—a $+5.8$ shift with non-overlapping 95% bootstrap CIs. On Undead 5×5, depth rises from 18.7 [16.6, 21.0] (SFT, n=38) to 20.7 [20.1, 21.3] (vanilla GSPO, n=324); the SFT estimate is noisy due to low solved counts but the direction matches Bridges. The depth allocation learned during puzzle RL surfaces in both held-out math and harder puzzle generalizations (Figure 12b).

Within-checkpoint, depth scales with task complexity. Holding the model fixed and varying the puzzle type (Figure 12a), Undead 5×5 elicits deeper chains than Bridges 8×8 in both SFT (14.4 vs 11.2) and vanilla GSPO (21.3 vs 20.0, averaged over all traces)—despite the smaller grid, ruling out the mechanical "bigger grid $\Rightarrow$ longer trace" story. We attribute this to the complexity of line-tracing under reflection in undead vs. the simpler bridge-count constraint.

Holding the model and puzzle type fixed and varying only the grid size on *solved* traces (Figure 12b, 95% bootstrap CIs), Bridges $7\times7 \rightarrow 8\times8$ raises mean chain depth by $+0.96$ for SFT but by $+6.63$ for vanilla GSPO ($6.9\times$ steeper slope); on Undead $4\times4 \rightarrow 5\times5$ the slopes are $+4.75$ vs $+7.47$. Vanilla GSPO is a *more aggressive depth-allocator* than SFT: when the task demands more reasoning, it stretches further. This is consistent with the RL stage training on harder puzzles than SFT, learning to compose longer compute–check chains under verifiable reward.

J. Solved-vs-unsolved chain-depth significance

Table 9 reports the per-checkpoint Mann–Whitney U test for the solved-vs-unsolved chain-depth split shown in Figure 6(b). The figure itself plots only the two means and the n-of-solved annotation; significance details are kept here for reference.

K. Training hyperparameters

All experiments use `allenai/OLMo-3-7B-Instruct-SFT` as the base. Both stages train LoRA adapters (rank 64, `all-linear`); stage-specific α values are in Tables 10–11.

Both runs use n_{gen}=8 on $8\times$B200 GPUs. All other RL hyperparameters are identical.

Verifier reward components. The exact term is binary: it is 1 iff the extracted `<answer>` verifies against the puzzle solution and 0 otherwise. The completion reward is a dense board-completion score used before exact solving is common: after strict shape checks, it computes the fraction of grid cells matching the ground-truth solution, gives cells that differ from the initial puzzle board weight 2, and raises the weighted fraction to power $p = 3$. The format reward is an XML-structure reward in $[0, 1]$: it gives 0.25 each for exactly one closed `<reasoning>` tag, exactly one closed `<answer>` tag, reasoning appearing before answer, and no duplicate opening tags. Thus the table's $0.1\cdot$format-reward contribution is at most 0.1 and is intended only to stabilize response formatting.

Table 8. Mean exploitation-motif counts per trace ($k = 3$: CHECK→COMPUTE→CHECK; $k = 5$: CHECK→COMPUTE→CHECK→COMPUTE→CHECK).

	$k = 3$ motif			$k = 5$ motif		
Checkpoint	Hard	Easy	AIME25	Hard	Easy	AIME25
Base	0.00	0.01	0.01	0.00	0.00	0.00
SFT	0.57	0.71	0.88	0.17	0.22	0.30
Vanilla GSPO	1.40	1.34	1.48	0.58	0.58	0.64
Novelty bonus	1.00	1.11	1.19	0.38	0.43	0.50

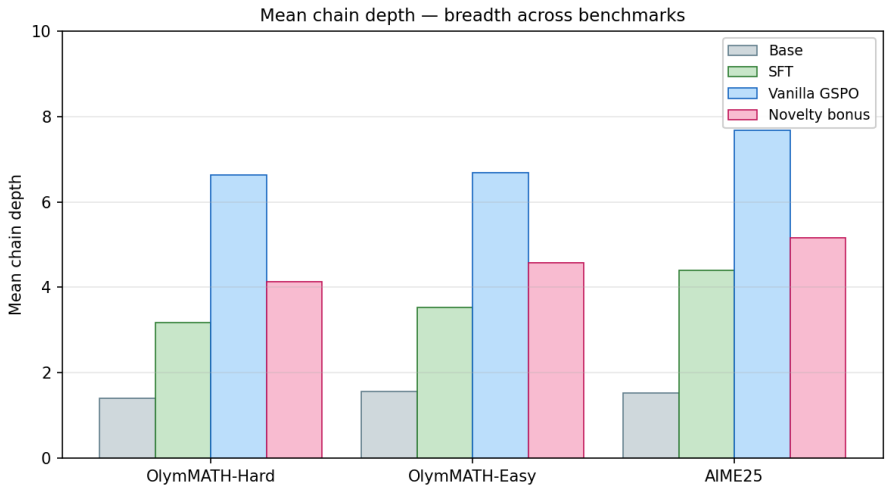


Figure 9. Mean chain depth across the four checkpoints on OlymMATH-Hard, OlymMATH-Easy, and AIME25.

L. Motif extraction details

Span segmentation algorithm. Each reasoning trace is processed as follows. First, the reasoning block is extracted from `<reasoning>` tags if present; otherwise the full response is used. The text is then split at paragraph boundaries (double newlines) and further split within each paragraph at discourse markers detected by a multi-line, case-insensitive regex covering: numbered steps (*Case N*, *Step N*, *Option N*); sequential connectives (*First*, *Second*, *Then*); hypothetical openers (*Suppose*, *Assume*, *Let's*); verification openers (*Check*, *Verify*, *Confirm*); self-correction markers (*Wait*, *Actually*, *Hmm*); redirection markers (*Instead*, *Alternatively*, *Going back*); conclusion markers (*Therefore*, *Hence*, *Thus*); and failure markers (*Contradiction*, *This doesn't work*). Short fragments below 80 tokens are merged into an accumulation buffer until the threshold is reached; long fragments above 250 tokens are split at sentence boundaries (period, exclamation mark, or question mark followed by whitespace). A trailing buffer below 40 tokens is merged into the previous span rather than emitted as a standalone fragment. Token counts during merging use a character-based estimate (character count divided by 4) for speed; the OLMo3 tokenizer is invoked only for the final token count on each emitted span.

Span classifier training. We fine-tune Qwen3-1.7B (Team, 2025) as a nine-way sequence classifier over the primitive vocabulary of Table 1. Training data consists of 35k judge-labeled spans: 17k from puzzle reasoning traces and 18k from mathematics reasoning traces. Labels were produced by a jury of LLM judges (DeepSeek-V3.2 and DeepSeek-Chat (Liu et al., 2025)), iterating prompt refinements until inter-judge agreement reached 86% on a held-out span set. Each judge labels a span by 5-sample self-consistency with a majority vote; the converged labeling prompt is reproduced verbatim in Appendix S. The resulting classifier reaches 0.80 macro-F1 against judge consensus on a held-out test set. All primitive sequences in Sections 4.1–5.1 use this classifier.

Sequence representation. For each reasoning trace, our classifier of Section 4 produces a sequence of primitive labels, one per detected episode. We restrict to the 8 reasoning labels (PLAN, SETUP, ENUMERATE, HYPOTHESIZE, COMPUTE, CHECK, BACKTRACK, SUMMARIZE) and treat OTHER as a boundary token `__BREAK__` that prevents adjacency across formatting/meta spans (e.g. `[CHECK, OTHER, SETUP]`→`[CHECK, __BREAK__, SETUP]`, so CHECK→SETUP

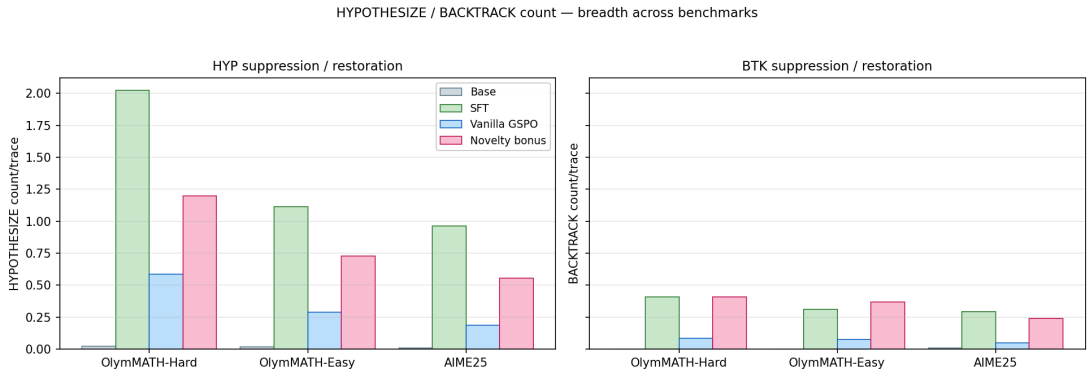


Figure 10. Mean HYPOTHESIZE and BACKTRACK counts per trace, faceted by benchmark.

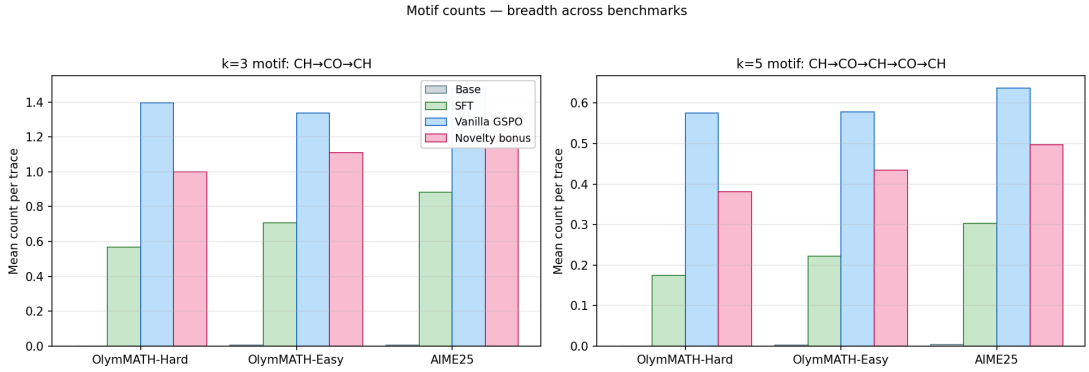


Figure 11. Mean $k = 3$ and $k = 5$ exploitation-motif counts per trace, faceted by benchmark.

is not counted as a bigram). OTHER is predominantly output formatting in our traces (91.5% of OTHER spans appear in the last 20% of the trace) and treating it as a reasoning token would inflate cross-section adjacencies.

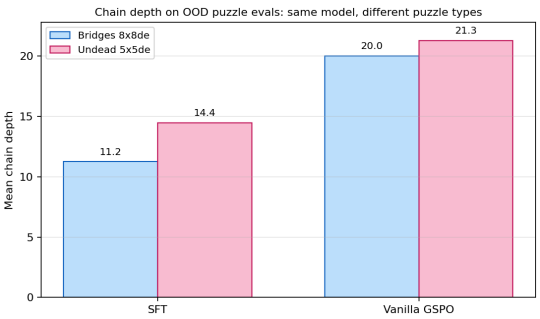
k-gram extraction. For each $k \in \{2, \ldots, 15\}$ we extract all length-k subsequences via a sliding window and count motifs that do not cross a __BREAK__. We use the observed vocabulary only (full enumeration is infeasible: $9^{10} \approx 3.5 \times 10^9$). Counts are length-normalised per trace as $\mathrm{freq}(m) = \mathrm{count}(m)/(L - k + 1)$ where L is the post-break sequence length, and motifs with raw count below 5 (for $k \leq 5$) or 10 (for $k > 5$) are filtered as noise.

Reporting. Section 4.2 reports per-trace counts for three concrete motifs (Figure 3b). For consistency with related work that does not distinguish overlapping vs non-overlapping matches, those counts use non-overlapping matches; cross-checkpoint comparisons are unaffected by the choice.

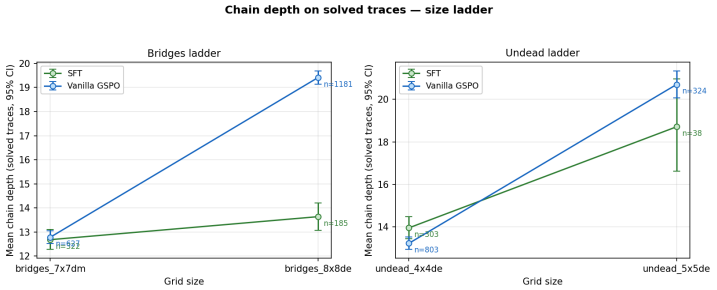
Motif categories used in Figure 2. The Section 4.1 motif row aggregates length-3 trigrams into three curated categories. Let $w = (w_1, w_2, w_3)$ be a length-3 window over the primitive sequence; let $\mathrm{ALLOWED_MIDDLE} = \{$PLAN, SETUP, ENUMERATE, COMPUTE, SUMMARIZE, OTHER$\}$. The three category predicates are:

- *Recovery*: HYPOTHESIZE $\in w$ **and** BACKTRACK $\in w$ (order-agnostic; matches any trigram window containing both recovery primitives).
- *Exploitation*: $w = ($COMPUTE, CHECK, COMPUTE$)$ (anchored on COMPUTE; the canonical compute↔check alternation chain).
- *Verification*: $w_1 =$ CHECK, $w_3 =$ CHECK, $w_2 \in \mathrm{ALLOWED_MIDDLE}$ (anchored on CHECK with a non-recovery, non-CHECK middle slot).

The middle-slot exclusions in Verification are deliberate. Excluding CHECK from w_2 prevents long runs of consecutive



(a) Same model, different puzzle type.



(b) Same model and puzzle type, different grid size (solved traces only; 95% CI).

Figure 12. Within-checkpoint depth scaling. Chain depth tracks task complexity at inference: **(left)** undead is deeper than bridges despite a smaller grid; **(right)** a one-step grid increase lifts chain depth far more under vanilla GSPO than under SFT. Vanilla GSPO is a more aggressive depth-allocator.

Table 9. OlymMATH-Hard chain depth, solved vs unsolved traces, per checkpoint. Mann–Whitney U tests are two-sided and unadjusted; sample sizes are n_{solved} and n_{unsolved} respectively, summing to the pass@32 trace pool (3,200 per checkpoint).

Checkpoint	$\bar{d}_{\text{solved}}$	$\bar{d}_{\text{unsolved}}$	Δ	U	p	n_{solved}
Base	–	0.83	–	–	–	0
SFT	7.35	6.06	+1.29	159,073	0.003	86
Vanilla GSPO	9.96	14.46	−4.50	121,328	0.020	91
Novelty bonus	10.29	8.27	+2.02	184,836	0.009	104

CHECKs from inflating the verification count via the sliding window without adding semantic content (a sticky-CHECK artefact rather than a check–act–check pattern). Excluding HYPOTHESIZE and BACKTRACK from w_2 keeps Verification disjoint from Recovery: a window like (CHECK, HYPOTHESIZE, CHECK) that would otherwise satisfy both predicates is counted under Recovery only. The Exploitation predicate is exact-match (COMPUTE-anchored), so it is also disjoint from the other two by construction. This three-way disjointness is what makes the three category bars in Figure 2 sum interpretably without double-counting. The $k = 5$ extension used elsewhere (CHECK→x→CHECK→y→CHECK with $x, y \in$ ALLOWED_MIDDLE for verification, and the analogous CHECK→COMPUTE→CHECK→COMPUTE→CHECK for exploitation) preserves the same anchor and disjointness rules.

M. Metric definitions

Let $s = (s_1, \ldots, s_L)$ denote the primitive sequence of a trace (post-OTHER-filtering as in Appendix L) and let T denote the trace's response token count.

Chain depth. We define an *exploit run* as a maximal contiguous block of primitives drawn from EXPLOIT $=$ {COMPUTE, CHECK, SETUP}. The chain depth of a trace is the length of its longest exploit run:

$$\text{depth}(s) \;=\; \max_{i \le j} \big\{\, j - i + 1 \;:\; s_i, \ldots, s_j \in \text{EXPLOIT} \,\big\}.$$

We include SETUP alongside COMPUTE and CHECK because it represents on-track implementation work (introducing notation, defining auxiliary variables) rather than strategic decomposition or branching; excluding it would shatter compute–check runs every time the model names a new variable. The ranking is robust to this choice: with EXPLOIT $=$ {COMPUTE, CHECK} (no SETUP), mean chain depths on OlymMATH-Hard are Base 0.83, SFT 2.27, vanilla GSPO 4.70, novelty 3.25 — the same Base $<$ SFT $<$ vanilla GSPO ordering with novelty between SFT and vanilla GSPO, and the SFT→vanilla GSPO "roughly doubles" claim (2.07× vs 2.09× with SETUP). We report mean depth and the 90-th percentile per checkpoint.

Table 10. Stage 1 (SFT) hyperparameters. The SFT corpus contains rejection-sampled DSR puzzle traces only; we keep traces whose final answer verifies against the puzzle scorer and discard the rest. Epoch 5 is selected as the SFT endpoint (best held-out puzzle pass@1).

Setting	Value
Base model	OLMo-3-7B-Instruct-SFT
LoRA rank / α / target	64 / 64 / `all-linear`
Optimizer / learning rate	AdamW / 2×10^{-4}
Train batch size	32 (8 GPUs $\times$ micro-batch 1×4 gradient-accum. steps)
Max sequence length	32,000 (corpus filtered to $\leq$ 28k tokens)
Epochs / endpoint	8 trained / epoch 5 used downstream
Precision	bfloat16 training; fp32 LoRA merge

Table 11. Stage 2 (RL) hyperparameters. The vanilla-GSPO baseline and the novelty-bonus variant share all of these settings; the only difference is whether the novelty bonus (Section 5, α=0.1, top-k=100, z-clip 2) is added on top of the GSPO reward.

Setting	Value
Init / KL reference	SFT epoch 5, fp32-merged checkpoint
LoRA rank / α / target	64 / 128 / `all-linear`
Optimizer / learning rate	AdamW / 5×10^{-5} (cosine, min ratio 0.3)
Train batch size	128 prompts
Rollouts per prompt	8
PPO mini-batch / epochs	1024 / 4
Sampling	T=0.8, top-p=1.0
Max prompt / response length	3,000 / 28,000 tokens
GSPO clip ratios	$\epsilon_{\text{low}}=2.5\times10^{-3}$, $\epsilon_{\text{high}}=5\times10^{-3}$
KL coefficient β	10^{-3} (low-variance KL)
Reward	exact + completion reward (p=3, changed-cell weight 2) + 0.1·format reward
Epochs over data / total steps	1 (no data recycling) / 70 steps
GPUs	8×B200

Mean exploit-run length. Let $r_1, r_2, \ldots, r_M$ be the lengths of the maximal exploit runs in s (a run is a maximal contiguous block of primitives in EXPLOIT). The mean exploit-run length of a trace is

$$\text{mean_run}(s) \;=\; \frac{1}{M}\sum_{m=1}^{M} r_m,$$

undefined when $M = 0$; we report mean over traces with $M \geq 1$. Unlike chain depth, which picks the single longest run, mean run length averages all exploit runs in a trace and is therefore sensitive to whether exploit primitives are clustered into a few long runs (high mean) or scattered as many short runs (low mean). We use it alongside chain depth to ensure the depth-of-exploitation claim is not an artefact of one long run with many scattered short runs around it.

N. Alternative diversity mechanisms

Production GSPO runs at our scale take roughly 70 wall-clock hours on 8×B200 GPUs per configuration (≈560 GPU-hours), making it impractical to iterate diversity-bonus designs at full setup. We therefore ran a smaller *mini* ablation setup—2 puzzles (Bridges and Undead) and a 12k-token response budget per rollout, against the 4-puzzle, 28k-token production setup—roughly 6× cheaper per run. We use mini to compare diversity-bonus candidates and run only the best at production scale. Other diversity-promoting interventions applied to the same setup either collapse or fall short on stability (Table 12).

At mini scale, the loss-level entropy bonus and our novelty bonus both reach `pass@32` $= 28.0\%$ at their best step—a tie on peak performance. The runs differ on training stability: the entropy-bonus run peaks at step 5 and degrades thereafter, while the novelty-bonus run trains stably through step 20. Entropy regularisation also does not target the specific HYPOTHESIZE/BACKTRACK subspace—it pulls the whole distribution toward uniformity. The entropy-advantage variants of Cheng et al. (2025) collapsed below baseline (17.0% no-KL, 14.0% with KL).

These pilots should be read as a screening study rather than a full-scale ranking of diversity mechanisms. The mini setup omits five puzzles and uses a shorter response budget, and we did not train every alternative at production scale. The evidence supports two narrower claims: first, the frozen-reference novelty bonus was the most stable candidate in our pilot

set; second, when scaled to the production setup, it improved over the vanilla-GSPO production baseline (36% vs 29% `pass@32`). A full production-scale comparison against entropy and trajectory-rarity baselines remains future work.

Table 12. Alternative diversity mechanisms on OlymMATH-Hard. All entries use GSPO at the same data and clip configuration; only the diversity mechanism differs. Numbers are best-step `pass@8` / `pass@32` over $n = 32$ rollouts.

Method	Scale	`pass@8`	`pass@32`
GSPO baseline	production	12.6	29.0
Loss-level entropy bonus (unstable)	mini	12.4	28.0
Entropy-advantage no-KL (Cheng et al., 2025)	mini	8.9	17.0
Entropy-advantage +KL	mini	5.5	14.0
Novelty bonus	mini	13.1	28.0
Novelty bonus (ours)	**production**	**16.4**	**36.0**

O. Novelty-bonus algorithm

Figure 13 gives the per-step computation that augments the GSPO reward with the frozen-reference novelty bonus described in Section 5. The bonus is reward shaping: it is computed once per rollout under the *frozen* SFT reference policy π_{ref} (no gradients), then added to the last valid token's reward before advantage computation. All other GSPO machinery (clip ratios, KL penalty, optimisation) is unchanged.

Top-k frozen-reference novelty bonus (per training step).
Inputs: rollouts $\{r_i\}_{i=1}^{B}$ grouped by prompt id $\mathrm{uid}(r_i)$; per-token frozen reference log-probs $\log \pi_{\text{ref}}(r_i)$; response masks m_i; correctness flags $a_i \in \{0, 1\}$; per-token rewards R_i; hyper-parameters α=0.1, k=100, z_{clip}=2.

1. For each rollout i, compute the masked per-token NLL $\mathrm{neg_lp}_i = -\log \pi_{\text{ref}}(r_i) \odot m_i$ and the top-k score $s_i = \frac{1}{k}\sum_{t \in \mathrm{TopK}_k(\mathrm{neg_lp}_i)} \mathrm{neg_lp}_{i,t}$.
2. For each prompt group g, let $C_g = \{i \in g : a_i = 1\}$ be the correct rollouts. Skip the group if $|C_g| < 2$ or $\mathrm{std}(\{s_i\}_{i \in C_g}) < 10^{-6}$.
3. Compute within-group statistics $\mu_g, \sigma_g = \mathrm{mean}, \mathrm{std}$ of $\{s_i\}_{i \in C_g}$, and set $z_i = \mathrm{clip}((s_i - \mu_g)/\sigma_g, -z_{\text{clip}}, +z_{\text{clip}})$ for each $i \in C_g$.
4. Add the bonus at the last valid token of each correct rollout: $R_i[\text{last valid}] \mathrel{+}= \alpha \cdot z_i$.
5. Compute GSPO advantages from the modified token-level rewards R as usual.

Figure 13. Pseudocode for the frozen-reference top-k novelty bonus described in Section 5. The bonus is reward shaping: it is computed once per rollout under the frozen SFT reference and added to the last valid token's reward before advantage computation. All other GSPO machinery (clip ratios, KL penalty, optimisation) is unchanged.

The bonus is applied only to correct rollouts ($a_i = 1$): incorrect rollouts already receive zero base reward, so adding novelty there would amount to rewarding incorrect novel reasoning. Within-group z-scoring with the z_{clip}=2 cap bounds $|\alpha z| \leq 0.2$, i.e. at most 20% of the maximum base reward, which keeps the bonus shaping rather than dominant.

P. Within-group signal analysis

The novelty bonus only has signal if SFT perplexity varies meaningfully across correct rollouts of the *same* prompt: with no within-group spread, the z-score in Figure 13 is undefined and the bonus collapses to zero. We measured this directly on 100 puzzle prompts (32 rollouts each, post-SFT) and on 100 OlymMATH-Hard prompts under both SFT and vanilla-GSPO checkpoints. Table 13 reports the mean within-prompt standard deviation of two candidate per-rollout signals: (i) mean per-token negative log-probability over the full response, and (ii) mean negative log-probability of the top-k most surprising tokens.

At a z-score cap of 2 and α=0.1, the mean-NLL signal would yield bonus magnitudes around $\pm 2 \times 0.024 \times 0.1 \approx 0.005$ on a base reward of order 1—below noise. The top-100 signal yields ≈ 0.05, which is where the bonus becomes a meaningful fraction of the base reward. We chose k=100 because the within-group std saturates between k=100 and k=200 (no further amplification), and smaller k becomes noisier across prompts of varying length.

Table 13. Within-prompt standard deviation (nats/token) of per-rollout SFT-NLL signals, averaged over 100 prompts. Mean NLL has near-zero spread: at our response lengths ($\sim$25k tokens), per-token NLL is dominated by formatting and connective tokens that share probability across all reasoning paths. Restricting to the top-k most surprising tokens isolates decision-critical positions and amplifies the within-group signal 10–15$\times$.

Signal	SFT puzzles	SFT math	Vanilla GSPO math
Mean NLL (full response)	0.024	0.024	0.024
Top-10 NLL	0.18	0.16	0.16
Top-100 NLL	0.276	0.260	0.245
Top-200 NLL	0.27	0.26	0.24

Q. Per-problem recovery split on OlymMATH-Hard

Section 5.1 focuses on the two diagnostic subsets for the selectivity claim: problems solved by both vanilla GSPO and the novelty-bonus variant, and problems solved only by the novelty-bonus variant. The full per-problem accounting over the 100 OlymMATH-Hard items is:

Table 14. Per-problem solve-set split used in Section 5.1. A problem is counted as solved by a checkpoint if at least one of its 32 rollouts verifies.

Outcome bucket	Number of problems
Solved by both vanilla GSPO and novelty bonus	20
Solved only by novelty bonus	16
Solved only by vanilla GSPO	9
Solved by neither	55

The primitive-count split shows the same selectivity pattern as the motif analysis in the body. On the shared problems, vanilla GSPO uses recovery primitives at a low rate (HYP 0.63, BTK 0.08 per trace), and the novelty bonus is higher but still modest (HYP 1.34, BTK 0.38). On unique-to-novelty problems, vanilla GSPO remains flat (HYP 0.63, BTK 0.10), while the novelty bonus rises (HYP 1.51, BTK 0.43); restricting to the actually solved novelty-bonus traces on these problems raises the rates further (HYP 2.25, BTK 0.46). Thus the recovered primitives are not deployed uniformly across all problems: they concentrate most strongly on the solved traces where vanilla GSPO fails.

R. Paired CoT inspection on OlymMATH-Hard

This appendix reproduces four worked OlymMATH-Hard rollouts to illustrate the mechanisms claimed quantitatively in the body. *Part A* (§4.2) pairs an SFT rollout against a vanilla-GSPO rollout where both are correct, to show that vanilla GSPO improves execution without changing the set of approaches. *Part B* (§5.1) pairs a vanilla-GSPO rollout that fails against a novelty-bonus rollout that succeeds, on problems vanilla GSPO never solves, to show that the novelty bonus's gain comes from in-trace recovery rather than from new mathematical machinery. Throughout, we score every rollout with `math_verify`, consistent with the scoring used across all main-paper evaluations.

Part A: Depth of execution (§4.2)

We pair, for the same problem, an SFT rollout and a vanilla-GSPO rollout that are both correct and share the same opening primitives. We pick two problems whose chain depths fall near each checkpoint's mean ($\bar{d} \approx 3.2$ for SFT, $\bar{d} \approx 6.6$ for vanilla GSPO).

In both cases the SFT and GSPO rollouts converge on the same algebraic reformulation in the same primitive sequence: single-variable substitution yielding a quadratic, then a vertex/endpoint range argument. They differ in how much compute–verify alternation follows: GSPO continues for one to three additional COMPUTE→CHECK cycles—numerical spot checks, boundary tests, sign discussion—before stopping, while SFT terminates after one verification pass. No new approach appears in the GSPO trace that is absent from the SFT trace. Figure 14 shows the primitive-sequence strips for both pairs; the longest COMPUTE/CHECK run (chain depth) is annotated.

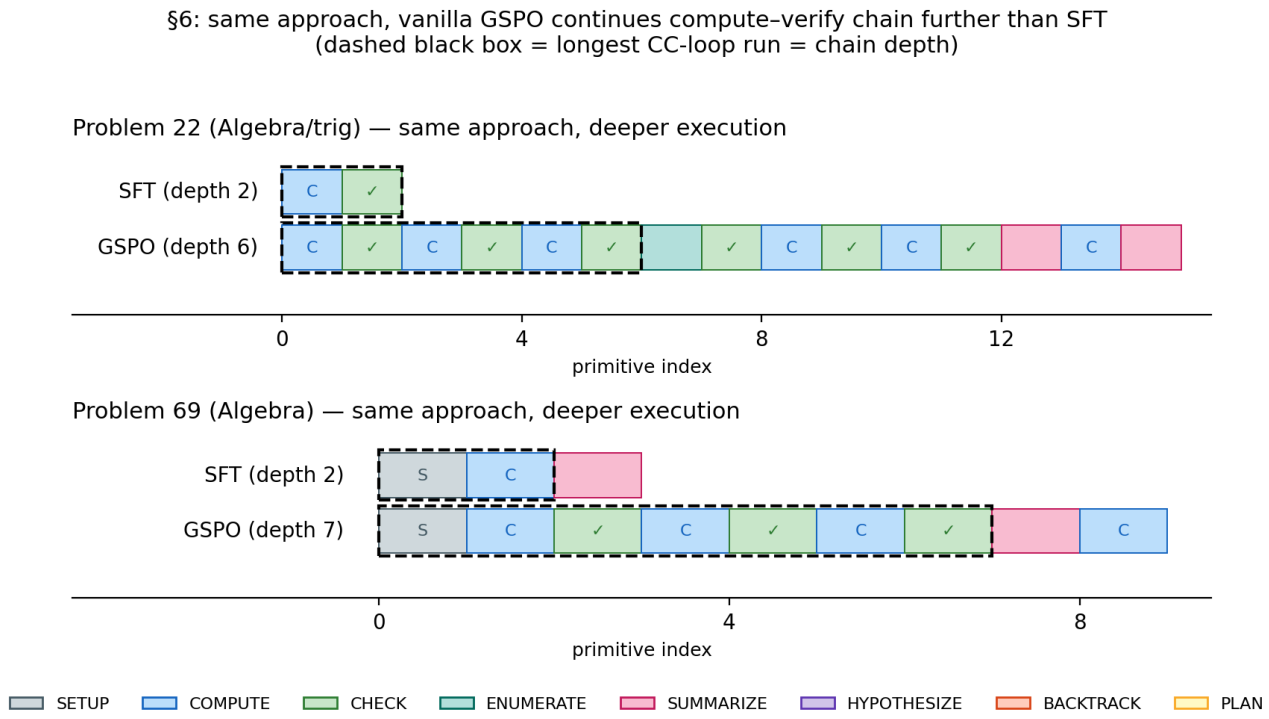


Figure 14. Primitive-sequence strips for the SFT vs vanilla-GSPO pairs on problems 22 and 69. Shared opening primitives sit at indices 0–1 in both pairs; the longest exploit run (dashed black box) is the chain-depth metric reported in §4.2.

Problem 1 (Algebra, trigonometric optimisation). Let $f(x) = \sin^4 \omega x - \sin \omega x \cos \omega x + \cos^4 \omega x$ with $\omega > 0$. If there exist $a, b \in [0, \pi]$ with $f(a) + f(b) = 9/4$, find the minimum ω. Gold answer: $\omega = 7/12$. SFT trace depth 2; vanilla-GSPO trace depth 6.

SFT (chain depth 2):
[COMPUTE] Use $\sin^4\theta + \cos^4\theta = 1 - 2\sin^2\theta\cos^2\theta$. Let $t = \sin\omega x \cos\omega x$, so $f(x) = 1 - 2t^2 - t$. With $t = \frac{1}{2}\sin(2\omega x)$ and $y = \sin(2\omega x) \in [-1, 1]$ rewrite $f(y) = 1 - \frac{1}{2}y^2 - \frac{1}{2}y$. Critical point at $y = -\frac{1}{2}$, giving $f = 11/8$; endpoints $f(\pm 1) = 0$.
[CHECK] So $\max f = 11/8$ and $2 \cdot 11/8 = 11/4 \geq 9/4$, hence $f(a) + f(b) = 9/4$ is achievable. Need $\sin 2\omega x = -\frac{1}{2}$, i.e. $2\omega x \in \{7\pi/6, 11\pi/6\} \pmod{2\pi}$. For $x \in [0, \pi]$ this requires $2\omega\pi \geq 7\pi/6$, hence $\omega \geq 7/12$. Minimum $\boxed{7/12}$.

Vanilla GSPO (chain depth 6):
[COMPUTE] Same substitution $t = \sin\omega x \cos\omega x$; same quadratic $g(t) = -2t^2 - t + 1$. Vertex at $t = -1/4$, $g(-1/4) = 9/8$; endpoints $g(\pm 1/2) \in \{0, 1\}$. Range of g on $[-1/2, 1/2]$ is $[0, 9/8]$.
[CHECK] To hit $f(a) + f(b) = 9/4$ the only option is $g(t_a) = g(t_b) = 9/8$, hence $t_a = t_b = -1/4$. Translating: $\sin 2\omega a = \sin 2\omega b = -\frac{1}{2}$, with $a, b \in [0, \pi]$ giving $\omega \geq 7/12$.
[COMPUTE] Solve $a = 7\pi/(12\omega)$ and $a = 11\pi/(12\omega)$ in $[0, \pi]$; the $11\pi/6$ solution forces $\omega \geq 11/12$, the $7\pi/6$ solution only $\omega \geq 7/12$. Tentative minimum $\omega = 11/12$ [...]
[CHECK] Re-derive at $b = \pi$, $\omega = 11/12$: first attempt gives $t_b = 1/4$ ("Wait, this contradicts earlier calculation. Let me recheck."). Recompute: $\sin(2\omega b) = \sin(11\pi/6) = -\sin(5\pi/6) = -\frac{1}{2}$, so $t_b = -1/4$ after all, $f(b) = 9/8$.
[COMPUTE] Re-examine the smaller bound $\omega = 7/12$: at this ω, $a = b = \pi$ gives $2\omega a = 7\pi/6$, $\sin = -\frac{1}{2}$, $t = -1/4$, $f(\pi) = 9/8$, so $f(a) + f(b) = 9/4$. The pair is allowed because the problem only asks for existence of $a, b \in [0, \pi]$.
[CHECK / ENUMERATE / CHECK / COMPUTE / CHECK / SUMMARIZE] Test $\omega = 1/2$ (below $7/12$): no $x \in [0, \pi]$ satisfies $\sin 2\omega x = -\frac{1}{2}$, confirming the bound is tight from below. Conclude $\boxed{7/12}$.

Both checkpoints reduce f to a single-variable quadratic and identify the vertex condition $\sin 2\omega x = -\frac{1}{2}$. SFT stops after one CHECK pass and asserts the answer; GSPO continues with two additional COMPUTE→CHECK cycles—a numerical self-correction at $b = \pi$ and an explicit lower-bound test at $\omega = 1/2$—before committing to $\omega = 7/12$.

Problem 2 (Algebra, range of a rational expression). If real x, y satisfy $x^2 - y^2 = 4$, find the range of $\frac{1}{x^2} - \frac{y}{x}$. Gold answer: $(-1, 1)$. SFT trace depth 2; vanilla-GSPO trace depth 7.

SFT (chain depth 2):
[SETUP] $x^2 - y^2 = 4 \Rightarrow x \neq 0$. Let $y = kx$; $x^2(1-k^2) = 4 \Rightarrow x^2 = 4/(1-k^2)$, requiring $|k| < 1$.
[COMPUTE] $E = 1/x^2 - y/x = (1-k^2)/4 - k = -(k^2+4k-1)/4$. Let $f(k) = k^2 + 4k - 1$. Vertex at $k = -2$ (outside $(-1, 1)$); $f(-1) = -4$, $f(1) = 4$, so on $(-1, 1)$ f ranges over $(-4, 4)$.
[SUMMARIZE] Hence $E = -f(k)/4$ has range $(-1, 1)$. (Earlier "± 1.25" boundary computations dismissed in one sentence as miscalculated.) $\boxed{(-1, 1)}$.

Vanilla GSPO (chain depth 7):
[SETUP] Note $x^2 - y^2 = (x-y)(x+y) = 4$; rewrite target as $(1-xy)/x^3$. Let $t = y/x \Rightarrow x^2 = 4/(1-t^2)$, forcing $|t| < 1$.
[COMPUTE] Target $= (1-t^2)/4 - t = (1-4t-t^2)/4 := f(t)$.
[CHECK] $f'(t) = -t/2 - 1$; $f' = 0$ at $t = -2$, outside $(-1, 1)$, so f is monotonic on $(-1, 1)$. Limits: $f(t \to 1^-) = -1$, $f(t \to -1^+) = 1$.
[COMPUTE] $f'(0) = -1$, so f is strictly decreasing on $(-1, 1)$; image is $(-1, 1)$ open. Endpoints not attained (would require $x^2 - y^2 = 0$).
[CHECK] Sign of x doesn't matter (both signs give same $f(t)$); $x = 0$ impossible.
[COMPUTE] Spot-check $t = 0$ ($x = \pm 2$, target $= 1/4$); $t = 0.5$ (target -0.3125, matches $f(0.5)$); $t = -0.5$ (target 0.6875, matches $f(-0.5)$).
[CHECK] Limit checks: $t = 0.99 \Rightarrow f \approx -0.985$ (approaches -1); $t = -0.99 \Rightarrow f \approx 1.245$. Reconcile: the supremum is approached only as $t \to -1^+$ and never attained; numeric at $t = -0.99$ is consistent with the decreasing-to-1 picture.
[SUMMARIZE / COMPUTE] Range $\boxed{(-1, 1)}$.

Both checkpoints adopt the substitution $t = y/x$, derive the same quadratic $f(t) = (1-4t-t^2)/4$ on $(-1, 1)$, and reach $(-1, 1)$. SFT terminates after one range argument and waves off a sketchy boundary computation as a miscalculation; GSPO continues with two additional COMPUTE→CHECK cycles—numerical spot-checks at $t \in \{0, \pm 0.5\}$ followed by limit checks at $t = \pm 0.99$—before committing.

Selection (Part A). Using `math_verify`, 13 OlymMATH-Hard problems have at least one correct rollout in both SFT ($81/3200$ traces) and vanilla GSPO ($89/3200$). Restricting to pairs that share the first two opening primitives and place the SFT depth in $[2, 4]$ and the GSPO depth in $[5, 9]$ yields 19 candidate pairs across 4 problems. The two reproduced above were chosen for cleanest opening match, depths near each checkpoint's mean, and algebraic-manipulation flavour, the dominant subject in the intersection. A small number-theory and combinatorics options exist (problems 11, 17, 25, 26, 91) but none paired typical-depth SFT and GSPO traces under the same-opening constraint, so they are not informative for the §4.2 depth-of-exploitation claim.

Part B: Productive recovery (§5.1)

Section 5.1 shows the novelty bonus's gain is concentrated on the 16 OlymMATH-Hard problems solved only by the novelty-bonus variant, and that its solved traces on these problems are enriched for recovery primitives (HYPOTHESIZE/BACKTRACK) that vanilla GSPO's failures do not show. Below we reproduce the underlying text for two such problems, chosen because the recovery primitive in the parquet's `primitive_sequence` aligns with a textually visible pivot. Figure 15 renders the four primitive sequences (vanilla-GSPO failures vs novelty-bonus successes on problems 4 and 73) as colour strips so the recovery pivot is visible at a glance. The common pattern: vanilla GSPO commits to a first answer that the trace itself flags as suspect; the novelty-bonus variant performs the same opening computation, then backtracks to a different conclusion.

Problem 3 (Algebra, multivariate optimisation). Let x, y, z be positive reals; find the minimum of $f(x, y, z) = \frac{(2+5y)(3x+z)(x+3y)(2z+5)}{xyz}$. Gold answer: $241 + 44\sqrt{30} \approx 481.6$. Of 32 vanilla-GSPO rollouts: 6 box 480 (a loose AM-GM bound), 3 box 784 (the symmetric ansatz), 9 truncate, the rest scattered. None match. The novelty-bonus variant's trace 3 succeeds (chain depth 2 for the GSPO failure; novelty trace $\approx 28{,}500$ chars with one HYPOTHESIZE and one BACKTRACK primitive at the pivot).

Vanilla GSPO failure (full short trace, $\approx 2{,}300$ chars):
[COMPUTE] Apply AM-GM to each numerator factor independently: $2+5y \geq 2\sqrt{10y}$, $3x+z \geq 2\sqrt{3xz}$, $x+3y \geq 2\sqrt{3xy}$,

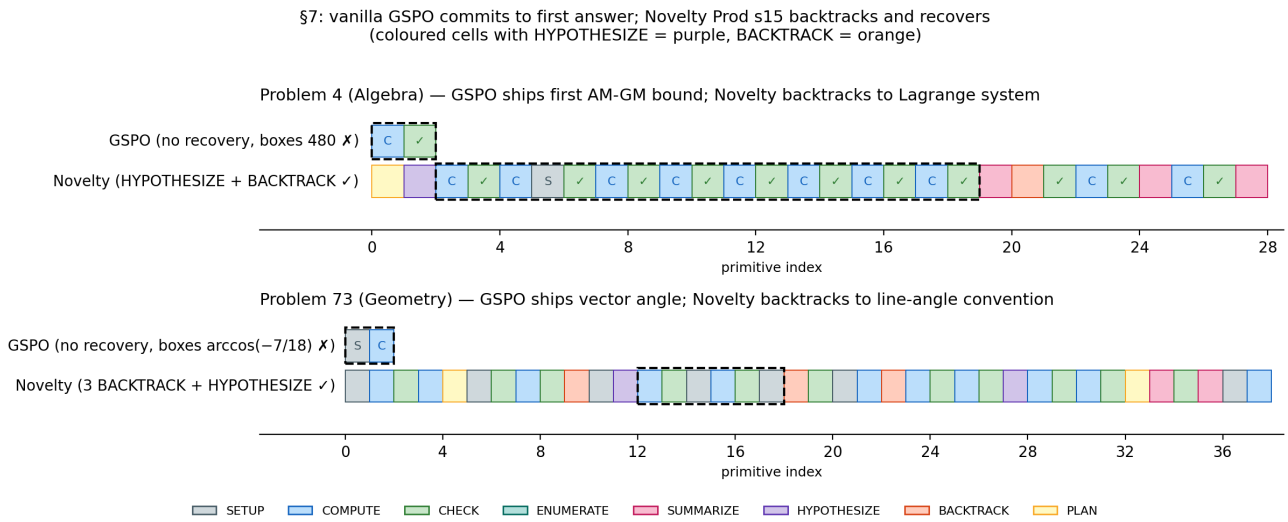


Figure 15. Primitive-sequence strips for the vanilla-GSPO vs novelty-bonus pairs on problems 4 and 73. The HYPOTHESIZE (purple) and BACKTRACK (orange) cells in the novelty rows mark the recovery pivots from the wrong first answer to the correct one; the GSPO rows have none.

$2z + 5 \geq 2\sqrt{10z}$. Multiply: numerator $\geq 16\sqrt{10y \cdot 3xz \cdot 3xy \cdot 10z} = 16 \cdot 30xyz = 480xyz$, hence $f \geq 480$.
[CHECK] "To check if equality holds, we need each AM-GM step tight: $2 = 5y \Rightarrow y = 2/5$; $3x = z$, $x = 3y \Rightarrow x = 6/5$, $z = 18/5$; $2z = 5 \Rightarrow z = 5/2$. This contradicts $z = 18/5 \neq 5/2$. Therefore the conditions cannot all be satisfied simultaneously, indicating equality might not be achievable. However, the AM-GM inequalities provide a lower bound, and since the function can approach this bound by choosing values close to those that satisfy the conditions, the minimum is indeed 480." $\boxed{480}$.

Novelty-bonus recovery:
[PLAN] Approach: try AM-GM on each factor; alternatively apply Lagrange multipliers to $\ln f$.
[HYPOTHESIZE] Try the symmetric ansatz $x = y = z = k$. The function reduces to $g(k) = 16(10k + 29 + 10/k)$, $k > 0$.
[COMPUTE → CHECK × 9] $g'(k) = 16(10 - 10/k^2) = 0 \Rightarrow k = 1$, second derivative positive, so $k = 1$ is a local min, $g(1) = 16 \cdot 49 = 784$. Spot-check $g(2) = g(0.5) = 864$, both higher.
[SUMMARIZE] "But is this the global minimum? We assumed $x = y = z$. Maybe there is a lower value when variables are not equal."
[BACKTRACK] Drop the symmetric ansatz. Use $F = \ln f$ and take $\partial F/\partial x = \partial F/\partial y = \partial F/\partial z = 0$ to get the actual critical-point system. Verify $x{=}y{=}z{=}1$ in equation (2): LHS $= 5/7 + 3/4 = 41/28 \neq 1$, so the symmetric ansatz is *not* a critical point—confirming the previous answer was wrong.
[CHECK → COMPUTE → CHECK → SUMMARIZE] Solve the critical-point system: it reduces to $x^2 = yz$, $15y^2 = 2x$, $2z^2 = 15x$. Substituting back gives $x = 1$, $y = \sqrt{2/15}$, $z = \sqrt{30}/2$. Plug in and simplify: $f_{\min} = 241 + 44\sqrt{30}$. $\boxed{241 + 44\sqrt{30}}$.

Both checkpoints reach for AM-GM first; vanilla GSPO multiplies the loose AM-GM bound, observes the equality conditions are inconsistent, and ships the bound anyway. Novelty starts on a stronger AM-GM (symmetric ansatz, giving 784), uses the partial-derivative system to verify whether the ansatz is a critical point, finds it is not, and *backtracks* to a Lagrange-style system whose explicit solution is the gold answer. The BACKTRACK primitive marks the moment Novelty abandons the ansatz; this is the exact step the GSPO trace is missing.

Problem 4 (Geometry, regular tetrahedron). For a regular tetrahedron $ABCD$, M, N are midpoints of edges AB, AC and P, Q are centroids of faces ACD, ABD. Find the angle between lines MP and NQ. Gold answer: $\arccos(7/18)$. Of 32 vanilla-GSPO rollouts: 14 box $\arccos(-7/18)$ (the angle between the *vectors*, not the lines), 3 box $90°$, 2 truncate, the rest miscellaneous. None match. The novelty-bonus variant's trace 25 succeeds with three BACKTRACK primitives.

Vanilla GSPO failure (full short trace, $\approx 2{,}900$ chars, no recovery primitives):
[SETUP → COMPUTE] Place $A = (1, 1, 1)$, $B = (1, -1, -1)$, $C = (-1, -1, 1)$, $D = (-1, 1, -1)$ (regular tetrahedron, edge $\sqrt{8}$). Midpoints $M = (1, 0, 0)$, $N = (0, 0, 1)$. Centroids $P = (-1/3, 1/3, 1/3)$, $Q = (1/3, 1/3, -1/3)$.

[COMPUTE] $\overrightarrow{MP} = (-4/3, 1/3, 1/3)$, $\overrightarrow{NQ} = (1/3, 1/3, -4/3)$. $\overrightarrow{MP} \cdot \overrightarrow{NQ} = -7/9$, $|\overrightarrow{MP}| = |\overrightarrow{NQ}| = \sqrt{2}$.
[CHECK / final commit] $\cos\theta = (-7/9)/2 = -7/18$. $\boxed{\arccos(-7/18)}$.

Novelty-bonus recovery (three BACKTRACK primitives, two HYPOTHESIZE):
[SETUP → COMPUTE → CHECK] Same coordinate system, same vectors, same $\cos\theta = -7/18$. Tentatively $\theta = \arccos(-7/18)$.
[COMPUTE / SETUP / CHECK] "However, the angle between two lines is defined as the smallest angle between their direction vectors. So if the calculated angle is obtuse, the actual angle between the lines is the acute supplement [...] $\cos(180^\circ - \theta) = +7/18$, so the angle between the lines is $\arccos(7/18)$. But the problem says 'the angle between MP and NQ'. Usually they mean the angle between the vectors as directed, but sometimes the smallest. Need to check."
[BACKTRACK #1] "Maybe I made a mistake in the coordinate scaling. Let me try a different system where the tetrahedron has edge length 2." → Resolve with vertices at $(1/\sqrt{2}, \pm, \pm, \pm)$. Same vectors up to uniform scale; $\cos\theta = -7/18$ again.
[BACKTRACK #2] Re-examine the problem statement: "P and Q are the centroids of faces ACD and ABD"—yes, what we used. → Recompute in original coordinates: still $-7/9$.
[BACKTRACK #3] Try a third coordinate system (edge length 1 with A at the origin). After one arithmetic error, confirm again that the *vector* angle has cosine $-7/18$.
[CHECK → HYPOTHESIZE → COMPUTE → CHECK / final commit] "In a regular tetrahedron, take two edges meeting at a vertex; the angle between them is $\arccos(1/3) \approx 54.7^\circ$, which is acute. So the convention takes the acute angle. Probably here also." $\boxed{\arccos(7/18)}$.

Both checkpoints compute the same vectors, the same dot product, and the same intermediate $\cos\theta = -7/18$. Vanilla GSPO commits to the obtuse vector angle the next segment. Novelty spends most of its length checking whether $-7/18$ is "really" the answer—testing two alternative coordinate systems (BACKTRACKs #1 and #3), re-examining the problem statement (BACKTRACK #2), and finally invoking the "angle-between-lines-is-the-smallest-angle" convention. Three BACKTRACK primitives mark three attempts to reconcile the numeric answer with the geometric question; the last produces the convention switch that yields the gold answer. The recovery here is on *interpretation*, not technique—the underlying coordinate-and-dot-product method is unchanged between the two traces.

Selection (Part B). The current per-problem split contains 16 OlymMATH-Hard problems solved only by the novelty-bonus variant (Appendix Q). Problems 4 and 73 were chosen from this set because the recovery primitive in the parquet aligns with a clean textual pivot (technique switch on 4; convention switch on 73) and the vanilla-GSPO failure traces are short enough to quote in full. We deliberately avoid using novelty-only winners whose successful traces contain no visible recovery primitive: in those cases the win comes from a clean first attempt under diverse seeds rather than from a recovery mechanism, and they would be misleading examples for the §5.1 claim.

S. Jury labeling prompt

The 35k seed spans were labeled by a jury of two DeepSeek judges—`deepseek-reasoner` (DeepSeek-V3.2) and `deepseek-chat` (Liu et al., 2025). Each judge classifies a single span into one of the nine primitives by 5-sample self-consistency, taking the majority-vote label; the per-span jury label is the consensus of the two judges, and we iterated prompt refinements until judge-to-judge agreement reached 86% on a held-out span set. The judge is shown the preceding context as background and asked to classify the span by the *function* it serves, returning a single-line JSON label (with an outcome field for CHECK spans). The converged prompt below is reproduced verbatim; the `{preceding_context}` and `{span_text}` placeholders are filled per span.

```
You are an expert annotator classifying a single span of a model's reasoning trace into one reasoning primitive
based on the FUNCTION the span plays in the reasoning. Do not classify by surface phrasing; classify by what the
span is trying to do. The trace may be on a logic puzzle or a math problem; the same taxonomy applies to both.

# Taxonomy (choose exactly one)
- PLAN: lays out a high-level strategy for SOLVING the problem -- choosing which approach to take, what order to
do things in, what the solver intends to do next. Forward-looking intent. Not the same as parsing or interpreting
the problem statement.
- SETUP: problem comprehension and working-representation building -- decoding a puzzle input grid, interpreting
clue notation, restating the problem's constraints in the model's own working terms, defining variables / unknowns
 for a math problem, cross-referencing the example to clarify format. The span is building WHAT the problem is,
before strategy or solving begins. Distinct from OTHER (truly off-topic content) and PLAN (forward strategy on HOW
 to solve).
- ENUMERATE: iterates over a finite collection of cases, possibilities, or configurations with TWO OR MORE
alternatives in scope, considered as a list. Examples: "Case 1...; Case 2...", "For p=2: ...; for p=3: ...; for p
=5: ...", "the row could be (a,b,c) or (b,a,c) or (c,a,b)".
- HYPOTHESIZE: posits ONE concrete tentative assumption or trial value (not a list of options) and explores its
consequences. If the span considers multiple alternatives, it's ENUMERATE.
- COMPUTE: performs concrete arithmetic, algebraic manipulation, LaTeX manipulation, substitution, simplification,
 or rule-based deduction. Produces a new value or expression as the output of a mechanical operation.
- CHECK: tests a previously-derived candidate, assignment, or sub-claim against a constraint or target. The act is
 evaluation -- outputting whether the candidate is consistent or inconsistent. The OUTCOME (pass/fail) is reported
 separately; both successful confirmations and detected contradictions are CHECK.
- BACKTRACK: abandons a current line of reasoning and pivots to a different approach or earlier state. The act of
switching paths (often follows a failed CHECK).
- SUMMARIZE: states a final answer or top-line conclusion that closes a chain of reasoning. The dominant act is *
announcing* the result, not deriving it. A span that mostly states "Therefore the answer is 42" or `\boxed{42}`
after the work is done is SUMMARIZE.
- OTHER: content that is NOT problem reasoning -- output formatting, JSON escaping, restating problem instructions
 verbatim with no working interpretation, transitional filler. If the span builds any working representation or
performs any reasoning act, prefer SETUP, PLAN, or one of the action labels.

# Tie-breakers (when a span contains multiple acts)
1. **SETUP vs PLAN:** if the span is decoding/interpreting the problem (e.g. "let's parse the grid: row 0 =
[...]", "the clue '1 2' means...", "let me define a = the number of red marbles, b = blue marbles"), label SETUP
-- building WHAT the problem is. PLAN is forward strategy on HOW to solve ("I'll start by...", "my approach is
...").
2. **SETUP vs OTHER:** if the span is doing real interpretive work on the problem's content, label SETUP. OTHER is
 for output formatting or pure instruction restatement that does not establish a working representation.
3. **HYPOTHESIZE vs ENUMERATE:** count alternatives in scope. One trial value = HYPOTHESIZE; >=2 listed =
ENUMERATE.
4. **HYPOTHESIZE vs COMPUTE:** a "Suppose X" or "By pigeonhole, let r be ..." that is a FORCED rule-based
consequence is COMPUTE -- it's a deduction, not a free assumption. HYPOTHESIZE is a free choice the model makes
for exploration ("Let me try x = 5...").
5. **CHECK vs COMPUTE:** if the span produces a new value via mechanical work, it's COMPUTE. If it tests an
existing value against a constraint, it's CHECK (regardless of outcome).
6. **CHECK vs ENUMERATE on per-case work:** when the dominant pattern is iterating over labeled cases ("For p=2:
...; For p=3: ...; For p=5: ..."), label ENUMERATE -- the listing is the point. Only label CHECK if the span tests
 ONE specific candidate.
7. **BACKTRACK vs CHECK:** a span that *finds* a contradiction is CHECK with outcome=fail. A span that *pivots* to
 a new approach is BACKTRACK. If pivoting is the dominant act, BACKTRACK.
8. **COMPUTE / CHECK vs SUMMARIZE at the end:** if the span derives the final number via real arithmetic and ends
with `\boxed{N}` or "the answer is N", the dominant act is the COMPUTE (or CHECK). SUMMARIZE is for spans that
mostly *announce* the final answer with little or no fresh derivation, often after the work was done in earlier
spans.

# Outcome field (for CHECK only)
For CHECK, also report the outcome of the test:
- pass     -- the candidate satisfied the constraint
- fail     -- the candidate failed (contradiction, inconsistency, mismatch)
- unclear  -- the test was inconclusive or aborted
For all other labels, set outcome to null.

Use the preceding context only as background; classify the SPAN.
```

```
# Examples (study the FUNCTION, not surface phrasing -- your span will differ in domain and wording)

## Example 1 -- PLAN
Preceding context: <<<Then we count the monster if: for V, only if direct; for G, only if reflected; for Z, always
.>>>
Span: <<<But note: The path might go through the same square multiple times from different directions, and each
time it might be direct or reflected depending on the segment. So we need to consider each crossing separately.
This is a bit complex. We might need to set up equations or constraints. Alternatively, we can try to solve by
logical deduction.>>>
Reasoning: states a forward-looking strategy ('we need to', 'we can try'); no derivation yet.
Output: {"label": "PLAN", "outcome": "null", "confidence": "high"}

## Example 2 -- SETUP
Preceding context: <<<three numbers formed by reading top to bottom is 99, we start by defining the digits in the
grid as follows: a,b,c / d,e,f>>>
Span: <<<We need to satisfy the following conditions:
1. The sum of the numbers formed by reading left to right: 100a + 10b + c + 100d + 10e + f = 999
2. The sum of the numbers formed by reading top to bottom: (100a + 10b + c) + (100d + 10e + f) = 99>>>
Reasoning: translates the problem statement into formal constraints in the model's working notation; no solving
yet.
Output: {"label": "SETUP", "outcome": "null", "confidence": "high"}

## Example 3 -- ENUMERATE
Preceding context: <<<Column 4: two separate filled cells, not adjacent.>>>
Span: <<<First, note that row 4 has two separate filled cells. That means in row 4, exactly two cells are filled,
and they are not adjacent. So the pattern in row 4 is like: F E F E, or F E E F, or E F E F. But note: if the
filled cells are at columns 1 and 3, that's F E F E. If at 1 and 4, that's F E E F. If at 2 and 4, that's E F E F
.>>>
Reasoning: lists three concrete alternative configurations for row 4; multiple alternatives in scope.
Output: {"label": "ENUMERATE", "outcome": "null", "confidence": "high"}

## Example 4 -- HYPOTHESIZE
Preceding context: <<<So clue should be 4. So that doesn't work.>>>
Span: <<<Maybe the rule is that Vampires are only visible if the line of sight to them does not involve any
reflection at all, meaning that the beam has not hit any mirror before reaching them. So if there is a mirror
before them in the path, they are not visible.>>>
Reasoning: posits one tentative rule ('Maybe the rule is...') as a free assumption to explore; no list of
alternatives.
Output: {"label": "HYPOTHESIZE", "outcome": "null", "confidence": "high"}

## Example 5 -- COMPUTE
Preceding context: <<<the losing positions are those where n == 2 mod 4.>>>
Span: <<<The sequence of losing positions is 2, 6, 10, 14, ..., which can be written as 4k + 2 for k >= 0. The
largest k such that 4k + 2 <= 2024 is found by solving 4k + 2 <= 2024: 4k <= 2022 => k <= 2022/4 = 505.5. The
largest integer k is 505, so the number of terms is 505 - 0 + 1 = 506.>>>
Reasoning: performs concrete arithmetic to derive a new value (506) via mechanical operations.
Output: {"label": "COMPUTE", "outcome": "null", "confidence": "high"}

## Example 6 -- CHECK
Preceding context: <<<Actually, (4,1) and (4,3) are not adjacent because col2 is between. So (4,3) is not
connected to the main group.>>>
Span: <<<So the pattern is split into two components: one containing (2,1),(3,1),(4,1),(2,2),(1,2),(2,4),(1,4) and
 the other containing just (4,3). That violates the global connectivity condition.>>>
Reasoning: tests the candidate placement against the connectivity constraint and reports it fails.
Output: {"label": "CHECK", "outcome": "null", "confidence": "high"}

## Example 7 -- BACKTRACK
Preceding context: <<<Contradiction.>>>
Span: <<<So if we try to not fix b=1 and c=1 from the island constraints, but wait, E has value 1, so b must be 1.
 F has value 1, so c must be 1. So they are forced. Thus, the only way to resolve is if d can be 3, but it can't.
So maybe the puzzle has a different set of connections. Perhaps there is an additional connection that provides an
 extra bridge to I.>>>
Reasoning: after a failed CHECK, abandons the current line and pivots to a new hypothesis ('maybe the puzzle has a
 different set of connections').
Output: {"label": "BACKTRACK", "outcome": "null", "confidence": "high"}

## Example 8 -- SUMMARIZE
Preceding context: <<<This results in intersections at points where x and y are multiples of 1/4 within the range
[0, 1].>>>
Span: <<<6. **Counting Intersections**: The intersections occur at (0,0), (1/4,1/4), (1/2,1/2), and (1,1). Thus,
the number of intersections is \boxed{4}.>>>
Reasoning: announces the final answer (4) after the work was already done; the boxed result is the dominant act.
Output: {"label": "SUMMARIZE", "outcome": "null", "confidence": "high"}

## Example 9 -- OTHER
Preceding context: <<<The user has provided a puzzle description.>>>
Span: <<<Let me format my final answer according to the instructions, putting the solution within <answer> tags as
 requested.>>>
Reasoning: output formatting / restating instructions; no problem reasoning, no working representation built.
Output: {"label": "OTHER", "outcome": "null", "confidence": "high"}
```

```
# Preceding context
<<<{preceding_context}>>>

# Span
<<<{span_text}>>>

# Output
Return only a JSON object on a single line:
{"label": "<one of PLAN|SETUP|ENUMERATE|HYPOTHESIZE|COMPUTE|CHECK|BACKTRACK|SUMMARIZE|OTHER>", "outcome": "<pass|
fail|unclear|null>", "confidence": "<high|medium|low>", "reasoning": "<one sentence>"}
```