Forging Obstacles into Reasoning via Guided Exploration

Shuyang Jiang[1,2,*], Wei Liu[2,*], Peixian Chen[2], Weibo Gu[2], Xing Sun[2], Qiang Lin[2],
Ya Zhang[3,†], Yanfeng Wang[3], Yu Wang[3,†]

[1]Fudan University [2]Tencent Youtu Lab [3]Shanghai Jiao Tong University

Group Relative Policy Optimization (GRPO) loses its learning signal on all-incorrect groups because every attempt receives the same reward. Yet 89.1% of the failed prefixes we audited contain enough mathematical progress to identify a concrete intermediate subproblem. Building on this finding, we propose `FORGE` (**F**orging **O**bstacles into **R**easoning via **G**uided **E**xploration), a self-guided reinforcement learning paradigm in which a single policy uses its own failed attempts to recover a learning signal on all-incorrect problems. The policy turns the impasses in its failed attempts into intermediate questions without access to reference solutions or target answers, then uses those questions to reattempt the original problem. Because several attempts can point to the same subproblem, *semantic selection* gives distinct subproblems priority over repeated formulations when choosing groups for solver training. When verification against the original answer identifies both correct and incorrect retries for a subgoal, the resulting reward contrast trains the solver and rewards the hinter for proposing that subgoal. In controlled 350-step comparisons from `Qwen3.5-4B-Base`, `FORGE` outperforms GRPO, DAPO, and reference-guided HATCH and SAGE in both Pass@1 and Pass@32. Using within 20K open-source problems and 500 training steps, `FORGE` surpasses the official post-trained `Qwen3.5-4B` by 5.05 percentage points in Pass@1 and 12.18 percentage points in Pass@32 across seven competition, Olympiad, and research-level mathematical benchmarks.

Correspondence: {ya_zhang,yuwangsjtu}@sjtu.edu.cn

1 Introduction

Reinforcement learning with verifiable rewards (RLVR) has driven substantial progress in mathematical reasoning for large language models [Jaech et al., 2024, Guo et al., 2025, Kimi Team et al., 2025, Hu et al., 2025, Zeng et al., 2025]. Group Relative Policy Optimization (GRPO) [Shao et al., 2024] and related critic-free policy-gradient methods [Ahmadian et al., 2024, Hu, 2025] derive their learning signal by contrasting verified outcomes across rollouts sampled for the same problem. When every sampled solution is incorrect, however, the group yields zero relative advantage and provides no outcome-based policy gradient. DAPO [Yu et al., 2025] uses dynamic sampling to replace uninformative groups, while experience replay reuses informative past rollouts [Zhan et al., 2025, Jiang et al., 2026]. Other methods draw on reference solutions or answers to guide new attempts [Li et al., 2026, Liu et al., 2025, Xie et al., 2026, Nath et al., 2025, Chen et al., 2026, Ding, 2026, Xia et al., 2026], as in HATCH [Wang et al., 2026] and SAGE [Liao et al., 2026]. This leaves open whether failed attempts themselves provide enough structure to recover the missing signal.

All-incorrect does not mean no progress. Figure 1a illustrates this with a cookie-passing rollout: it correctly accounts for one transfer, yet reaches a reasoning impasse on repeated transfers and answers incorrectly. An extracted subgoal isolating the opposite-seat configuration yields 1/16 correct in fresh guided attempts, versus 0/16 unassisted. A source-blinded audit of 250 failed prefixes finds extractable bottlenecks in **89.1**% of the candidate pool (Figure 1b), driven primarily by midway impasses (**71.6**%). We therefore view failures in all-incorrect groups as *recoverable reasoning impasses* whose partial derivations suggest a concrete subproblem to attempt next.

[*]Equal contribution
[†]Corresponding authors



Figure 1. `FORGE` **turns reasoning impasses into learning signals and stronger unassisted performance. (a)** A real failed rollout supplies a subgoal that yields 1/16 correct in fresh guided attempts on a 0/16 problem. **(b)** Source-blinded audit of candidate-pool bottlenecks and midway-impasse subgoal quality (left two plots), followed by increasing guided unlocking and cross-epoch unassisted solving on Epoch-1 all-incorrect problems (right two). **(c)** Downstream evaluation: validation trajectory on BeyondAIME against GRPO (left), scaled Pass@1 and Pass@32 (middle), and in-band accuracy across response-length intervals relative to Base (right). Gray and amber brackets compare `FORGE` with the official model and Base, respectively (*** $p < 0.001$; ns: $p \geq 0.05$).

We therefore propose `FORGE` (**F**orging **O**bstacles into **R**easoning via **G**uided **E**xploration), a single-policy reinforcement learning loop that turns this recoverable progress into a learning signal using only standard question–answer supervision $(q, a^\star)$ (Figure 2). Through *reference-free subgoal generation* (§3.1), the policy acts as a hinter to extract question-only subgoals from failed prefixes, without access to reference solutions or $a^\star$. *Semantic selection* (§3.2) distributes solver rollouts across distinct subgoals, exploring complementary routes past the bottleneck. An *unlocking reward* (§3.3) credits subgoals that restore mixed-outcome groups, allowing hinter and solver roles to train jointly within the same policy. During training, the fraction of subgoal-guided groups with mixed outcomes more than doubles (Figure 1b). By the third epoch, 69.2% (2,432/3,513) of Epoch-1 all-incorrect problems have been solved unassisted upon revisit, showing that successful solving extends beyond the guided attempts.

Starting from `Qwen3.5-4B-Base`, we evaluate `FORGE` across seven competition, Olympiad, and research-level mathematical benchmarks (Figure 1c). Under identical data and context budgets, `FORGE` learns substantially

faster than GRPO, surpassing GRPO's 350-step validation accuracy immediately upon entering the 32K stage, and outperforms GRPO, DAPO, and reference-guided HATCH and SAGE at the matched checkpoint (§4.2). With progressive context scaling within 20K open-source problems and 500 training steps, `FORGE` reaches **58.4**% Pass@1 and **83.3**% Pass@32, surpassing the official post-trained `Qwen3.5-4B` in both metrics. Relative to Base, it significantly improves single-attempt accuracy while preserving broad problem coverage under repeated sampling, with consistent accuracy gains across all response-length intervals from short solutions ($\leq$ 16K) to long-horizon reasoning (48–80K).

To conclude, our main contributions are three-fold:

- **Recoverable Reasoning Impasses:** Our diagnostic audit shows that failed rollouts frequently reach recoverable intermediate impasses that expose actionable subproblems, providing a basis for learning from all-incorrect problems.
- **Single-Policy Self-Guided RL:** `FORGE` combines reference-free subgoal generation, semantic selection, and an unlocking reward to jointly train hinter and solver roles from recovered learning signals.
- **Stronger Accuracy and Exploration Breadth:** `FORGE` outperforms controlled RL baselines and the official post-trained model in both single-attempt accuracy and multi-sample coverage, while consistently improving over the base model across all generation-length regimes.

2 Preliminaries

Group Relative Policy Optimization (GRPO). GRPO [Shao et al., 2024] adapts proximal policy optimization [Schulman et al., 2017] by estimating policy gradients from relative reward differences within a group of sampled responses to the same prompt, eliminating the need for a separate value network [Ahmadian et al., 2024]. For a math problem $q \sim \mathcal{D}$ paired with a verifiable ground-truth answer $a^\star$, the rollout policy $\pi_{\theta_{\text{old}}}$ samples a group of G responses $\{o_i\}_{i=1}^{G}$, each receiving a binary outcome reward $r_i = r^{\text{solve}}(q, o_i) = \mathbb{I}(\text{Verify}(o_i, a^\star) = 1) \in \{0, 1\}$ with empirical group pass rate $p_{\text{nat}}(q) = \bar{r} = \frac{1}{G}\sum_{j=1}^{G} r_j$. Under outcome supervision, all tokens $t \in \{1, \ldots, |o_i|\}$ in response o_i share the group-normalized advantage

$$\hat{A}_{i,t} = \frac{r_i - \bar{r}}{\sigma_r + \delta}, \qquad \bar{r} = \frac{1}{G}\sum_{j=1}^{G} r_j, \qquad \sigma_r = \sqrt{\frac{1}{G}\sum_{j=1}^{G}(r_j - \bar{r})^2}, \tag{1}$$

where $\delta > 0$ is a small constant for numerical stability. Let $h_{i,t} = (q, o_{i,<t})$ denote the token context prior to step t, and define the token-level importance sampling ratio

$$\rho_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t} \mid h_{i,t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid h_{i,t})}. \tag{2}$$

GRPO maximizes the token-level clipped surrogate objective

$$\begin{aligned} \mathcal{J}_{\text{GRPO}}(\theta) &= \mathbb{E}_{\substack{q\sim\mathcal{D} \\ \{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)}} \left[\frac{1}{\sum_{i=1}^{G}|o_i|} \sum_{i=1}^{G}\sum_{t=1}^{|o_i|} \ell_{i,t}(\theta) \right], \\ \ell_{i,t}(\theta) &= \min\Big\{ \rho_{i,t}(\theta)\hat{A}_{i,t},\ \text{clip}\Big(\rho_{i,t}(\theta), 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}}\Big)\hat{A}_{i,t} \Big\}, \end{aligned} \tag{3}$$

where ϵ_{low} and ϵ_{high} denote the lower and upper clipping thresholds. Following recent large-scale RLVR practice [Yu et al., 2025, Liu et al., 2025, Hu et al., 2025, Yue et al., 2025], we aggregate the policy loss at the token level and omit KL regularization ($\beta = 0$).

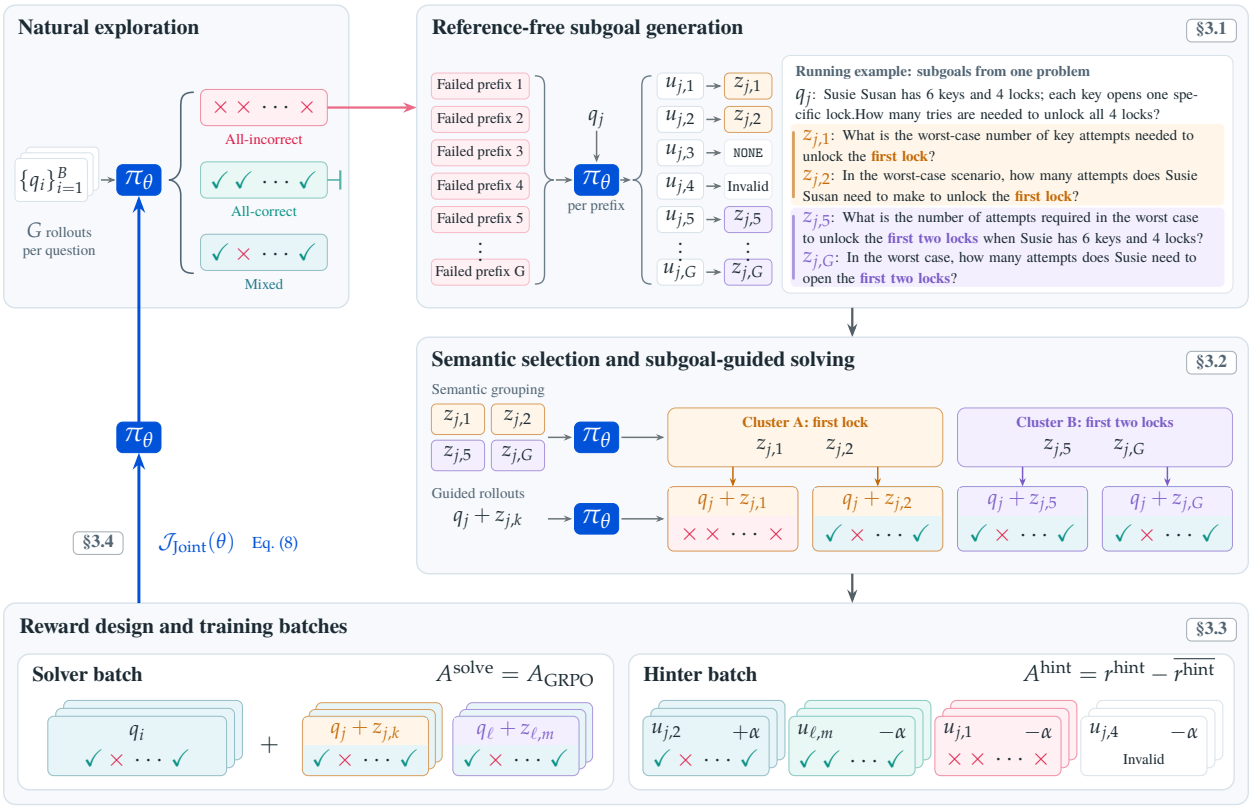

Figure 2. Overview of FORGE training paradigm. Natural exploration: Sampling G rollouts per problem q_i with π_θ discards all-correct groups, retains natural mixed groups, and routes all-incorrect problems to subgoal extraction. **Reference-free subgoal generation** (§3.1): From q_j and its G failed prefixes, π_θ samples $u_{j,1}, \ldots, u_{j,G}$ without $a^\star$, yielding subgoals $z_{j,k}$, invalid outputs when the outputs do not follow pre-defined formats, or NONE when the policy deems no available subgoals. **Semantic selection and subgoal-guided solving** (§3.2): π_θ clusters valid subgoals and samples guided rollouts on $q_j + z_{j,k}$ across distinct clusters. **Reward design and training batches** (§3.3): Natural and selected guided mixed groups form the solver batch ($A^{\text{solve}} = A_{\text{GRPO}}$); hinter outputs receive $+\alpha$ (mixed) or $-\alpha$ (uniform/invalid) under $A^{\text{hint}} = r^{\text{hint}} - \overline{r^{\text{hint}}}$ for the **joint policy gradient** update (§3.4).

Vanishing gradients on hard problems. Any prompt whose G sampled responses receive identical binary outcomes ($r_1 = \cdots = r_G = \bar{r}$) yields $\sigma_r = 0$, resulting in

$$\hat{A}_{i,t} = 0, \qquad \ell_{i,t}(\theta) = 0, \qquad \nabla_\theta \ell_{i,t}(\theta) = 0, \quad \forall\, i, t. \tag{4}$$

While all-correct groups ($p_{\text{nat}}(q) = 1$) represent problems the policy has already mastered, **all-incorrect** groups ($p_{\text{nat}}(q) = 0$) correspond to frontier problems where the model establishes valid initial setups and partial derivations (91.9% in our pilot audit; Figure 1b) yet reaches a reasoning impasse at an intermediate bottleneck without receiving a gradient signal. Simply discarding these all-incorrect groups via dynamic filtering [Yu et al., 2025] forfeits learning on the hardest problems in the dataset, whereas existing hint-based remedies [Liao et al., 2026, Xia et al., 2026, Wang et al., 2026] require privileged human- or teacher-written reference solutions to synthesize hints.

3 FORGE

FORGE uses a *single shared policy* π_θ for subgoal generation (**Hinter**) and subgoal-guided solving (**Solver**), learning from all-incorrect problems with only standard question–answer supervision $(q, a^\star)$. Figure 2

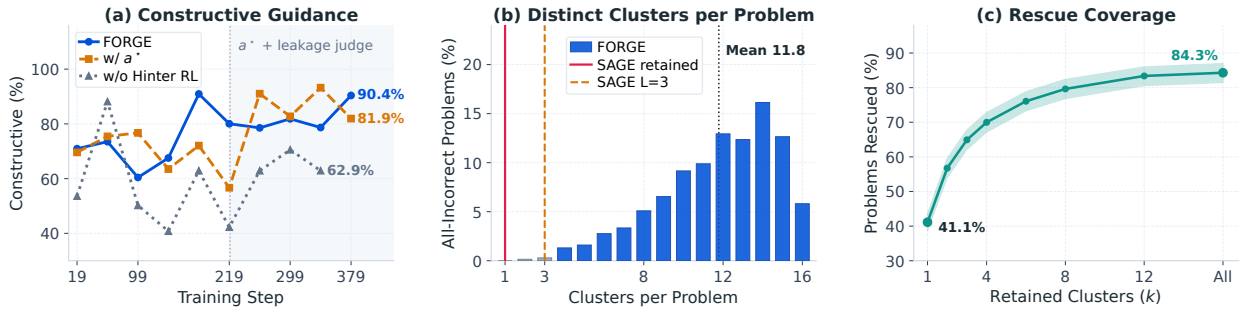

Figure 3. Reference-free joint RL sustains constructive subgoals, and problem-adaptive cluster diversity doubles rescue coverage. (a) Quality without answer shortcuts: Reference-free joint RL maintains high constructive subgoal quality throughout training, outperforming both solver-only updates (*w/o Hinter RL*) and answer-anchored extraction (*w/* $a^\star$), even after the latter adds an online leakage judge (shaded). **(b) Problem-adaptive subgoal breadth:** Unlike SAGE, which anchors hints to one reference solution (1 retained hint, up to $L = 3$ levels), `FORGE` extracts a problem-dependent number of distinct semantic clusters from failed traces, reflecting the diverse bottlenecks each problem exposes. **(c) Recovering the solver's gradient signal.** For all-incorrect problems with multiple subgoal clusters, retaining all clusters instead of one more than doubles the chance of restoring the gradient signal for the solver (shaded 95% bootstrap CI).

summarizes the learning paradigm, built around three central design choices: (1) *Reference-free subgoal generation*: the **Hinter** extracts question-only subgoals from multiple failed reasoning traces without viewing reference solutions or $a^\star$ (§3.1); (2) *Semantic selection*: after grouping candidate subgoals into semantic clusters and reattempting the original problem with the **Solver**, selection prioritizes distinct mathematical subproblems among mixed-outcome groups (§3.2); and (3) *Outcome-based hinter rewards*: original-answer verification rewards solver correctness and gives the hinter a uniform positive reward for producing mixed guided groups, with negative feedback for uniform outcomes or invalid format (§3.3). These mechanisms are trained together through a joint policy update (§3.4), and we theoretically analyze why rescued backfilling unlocks all-incorrect problems more effectively than natural backfilling (§3.5).

3.1 Reference-Free Subgoal Generation

Failure-grounded, question-only subgoal extraction. Whenever a natural rollout group $\{o_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid q)$ finishes with an all-incorrect outcome ($p_{\text{nat}}(q) = 0$), `FORGE` immediately invokes the live policy $\pi_{\theta_{\text{old}}}$ in its **Hinter** role to extract intermediate subgoals from the failed attempts. Unlike prior self-hinting frameworks [Liao et al., 2026, Xia et al., 2026, Wang et al., 2026], which condition hint generation on privileged reference solutions or answer-conditioned extraction schemes exposing the ground-truth label $a^\star$, our hinter receives *only* the original problem statement q and the chain-of-thought prefix $\tau_i = \text{think}(o_i)$ extracted from the i-th failed rollout. This design exploits a key empirical property of all-incorrect rollouts: in our source-blinded prefix audit (Figure 1b; Table 6 in Appendix C), **91.9**% of failed attempts already establish a valid mathematical setup and **89.1**% expose a concrete, extractable intermediate bottleneck where reasoning reaches an impasse. Conditioned on the extraction prompt $x_i^{\text{hint}} = \mathcal{P}_{\text{hint}}(q, \tau_i)$ (detailed in Appendix B), the policy samples an extraction trajectory $u_i \sim \pi_{\theta_{\text{old}}}(\cdot \mid x_i^{\text{hint}})$ across each of the G failed rollouts in parallel. Each completion u_i outputs either a single subproblem statement z_i formatted as `SUBPROBLEM: <question>`, an invalid output violating the pre-defined formats, or the token `NONE` when no constructive subproblem can be formulated. We separate valid subgoal extractions from format-invalid trajectories $\mathcal{U}_{\text{inv}}(q)$ and valid `NONE` abstentions; invalid outputs receive negative hinter feedback, while abstentions are excluded from guided solving and hinter updates (validation rules appear in Appendix A.1).

Answer anchoring favors restatement over decomposition. Unlike prior approaches, FORGE withholds both $a^\star$ and reference solutions during subgoal generation. Exposing the target answer creates a reward shortcut: the hinter can formulate a near-target question or superficial restatement that unlocks the group without identifying the intermediate reasoning step the solver lacks. Our longitudinal semantic audit confirms this failure mode (Figure 3a; protocol in Appendix A.4 and full breakdown in Table 8): without online filtering, answer-anchored joint RL ($w/\ a^\star$) steadily loses constructive quality as restatements and answer-leakage shortcuts accumulate (Figure 9a; concrete examples in Appendix E.2), allowing the hinter to collect positive rewards without learning genuine decomposition.

Online leakage filtering degrades as reasoning lengthens. A natural remedy is to prompt the live policy as an online judge to filter out subgoals that leak the ground-truth answer. While adding an online leakage filter temporarily lifts the answer-anchored constructive share (shaded region of Figure 3a), the judge's reliability remains coupled to the evolving policy. As training progresses and reasoning chains lengthen, judge verifications increasingly hit token-budget truncation (Figure 9b), eroding filtering precision so that the filtered pool falls back to 81.9% constructive guidance while FORGE climbs to **90.4**% (category breakdown in Figure 10). By removing the answer shortcut at its source, reference-free extraction allows direct hinter feedback to reward genuine decomposition without auxiliary verification overhead.

3.2 Subgoal-Guided Solving and Semantic Curriculum Selection

Semantic grouping of candidate subgoals. Because the G failed attempts on problem q may either reformulate the same intermediate question or pursue genuinely different constructions, candidate count alone does not measure the diversity of extracted subgoals. After collapsing exact duplicates among valid subgoals, the live policy groups candidates into semantic clusters $\{\mathcal{C}_1, \ldots, \mathcal{C}_M\}$ using the prompt in Appendix B. Candidates belong to the same cluster only when they ask the same mathematical question or proof obligation under the same assumptions, identifying distinct mathematical entry points for guided solving and curriculum selection. Let $\{\hat{z}_{q,k}\}_{k=1}^{K_q}$ denote the candidates submitted for guided solving, each retaining its cluster label and extraction trajectory $u_{q,k}$; multiple candidates may belong to one cluster (candidate-budget allocation is specified in Appendix A.1).

Subgoal-guided rollout. For each surviving subgoal candidate $\hat{z}_{q,k}$ ($k \in \{1, \ldots, K_q\}$), we construct a subgoal-guided prompt $\tilde{q}_{q,k} = \mathcal{P}_{\text{solve}}(q, \hat{z}_{q,k})$ by preserving the original problem q verbatim and appending the subgoal as an explicit reasoning scaffold:

```
{q}\n\n(Hint: Try to solve the following subproblem first; it may help solve the
problem above: {ẑ_{q,k}})
```

Within the same rollout step, the rollout policy $\pi_{\theta_{\text{old}}}$ samples a group of G guided solving trajectories $\{\tilde{o}_{q,k,j}\}_{j=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid \tilde{q}_{q,k})$. Eliminating separate pre-rollout probing phases allows FORGE to evaluate subgoal utility directly from the training rollouts themselves, maximizing GPU inference throughput. To prevent recursive curriculum drift, subgoal-guided groups are never used as sources for further subgoal extraction.

Semantic coverage in the training batch. Each step targets B solver prompt groups of G trajectories each. We first retain all $B_{\text{nat}} = |\mathcal{G}_{\text{nat}}^{\text{mix}}|$ natural mixed groups, whose empirical pass rates satisfy $0 < p_{\text{nat}}(q) < 1$. The remaining $B_{\text{rem}} = B - B_{\text{nat}}$ slots admit guided groups from $\mathcal{G}_{\text{guide}}^{\text{mix}}$, the pool with $0 < p_{\text{guide}}(q, \hat{z}_{q,k}) < 1$.

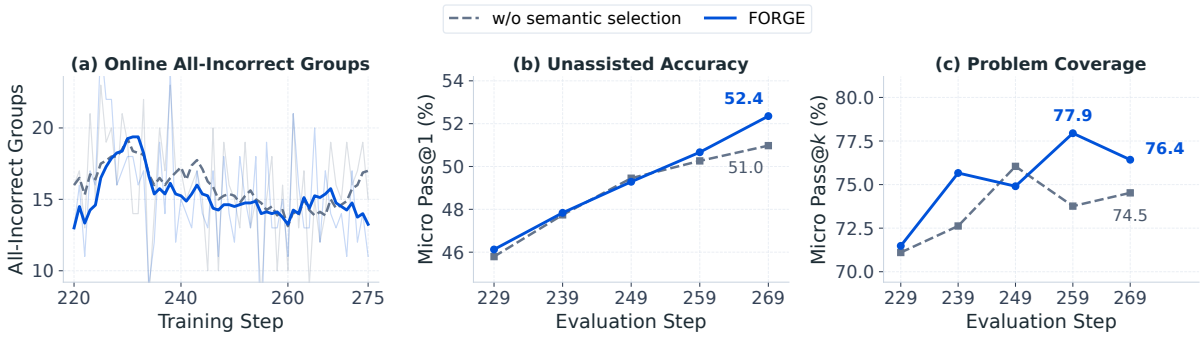

Figure 4. Semantic curriculum selection accelerates frontier unlocking and improves both unassisted accuracy and coverage. Starting from the same mid-training checkpoint under an identical batch budget, FORGE prioritizes distinct subgoal clusters before revisiting any selected cluster, compared against cluster-agnostic selection (*w/o semantic selection*). **(a) Faster online unlocking:** Allocating slots across diverse clusters leaves fewer natural training groups all-incorrect per step (bold: 8-step trailing average). **(b) Higher single-attempt accuracy:** On periodic 32K held-out evaluations, FORGE steadily widens its micro-average Pass@1 lead. **(c) Broader problem coverage:** Training on complementary entry points expands the set of solvable problems, lifting micro-average Pass@k.

Selecting guided groups solely by pass rate can repeatedly admit one semantic cluster while leaving another unrepresented. We therefore **prioritize distinct semantic clusters before admitting additional groups from any cluster**. Within each cluster, mixed groups are ordered by increasing p_{guide}; selection proceeds in rounds over source problems, taking the first group from each available cluster before considering its second group, and so on, until the budget is filled. The coverage priority applies across all source problems, so a repeated cluster cannot displace an uncovered one merely because its next group has a lower pass rate. Every admitted group retains all G trajectories; any residual slots are filled with remaining natural groups to form $\mathcal{B}_{\text{solve}}$ with $N_{\text{solve}} = B \times G$. Appendix A.2 gives the detailed ordering procedure.

Diverse semantic clusters unlock complementary entry points on all-incorrect problems. A single reference solution anchors hint generation to one predefined solution path: SAGE [Liao et al., 2026] supplies one shared hint per retained solver group and probes up to $L = 3$ progressive levels along that path. Extracting from multiple failed thinking prefixes instead exposes different attempted constructions and unresolved subproblems. On audited all-incorrect problems, FORGE evaluates an average of **11.8** distinct semantic clusters per problem, exceeding three clusters in nearly every case (Figure 3b). This breadth directly strengthens problem unlocking: retaining all clusters more than doubles mixed-group rescue coverage compared with a single cluster (**41.1**% $\rightarrow$ **84.3**%; Figure 3c). On the 4×4 board-counting problem in Appendix E.1, for instance, a subgoal asking for the 2×2 case unlocks the original problem while all eight other clusters—including one that counts admissible rows—remain all-incorrect.

Training across semantic groups improves solver learning. Selecting by semantic cluster allocates the fixed solver batch across distinct mathematical entry points rather than reinforcing equivalent reformulations. To isolate this effect under an identical sampling and optimization budget, we branch from the same mid-training checkpoint and compare FORGE with a cluster-agnostic baseline that rotates over source problems by lowest pass rate (*w/o semantic selection*). Semantic selection achieves complete cluster coverage while sharply reducing within-cluster repeats, yielding three consistent gains on unassisted solving (Figure 4): fewer natural training groups remain all-incorrect online, and periodic 32K evaluations show clear leads in both single-sample accuracy (+**1.4** pp micro Pass@1) and problem coverage (+**1.9** pp micro Pass@k). Full batch statistics and per-checkpoint results appear in Table 13.

3.3 Solver and Hinter Reward Design

Reward correct solutions and learnable guidance. The two roles receive feedback from the same original-answer verifier, but optimize different behaviors: the solver is rewarded for solving the problem, while the hinter is rewarded for making an all-incorrect problem yield a mixed group with a nonzero GRPO learning signal. All natural and guided solver responses are graded against the original answer $a^\star$:

$$r^{\text{solve}}(q,o) = \mathbb{I}(\text{Verify}(o,a^\star)=1) \in \{0,1\}, \qquad p_{\text{guide}}(q,\hat{z}_{q,k}) = \frac{1}{G}\sum_{j=1}^{G} r^{\text{solve}}(q,\tilde{o}_{q,k,j}). \tag{5}$$

Uniform reward for mixed guided groups. A guided group with $0 < p_{\text{guide}} < 1$ contains both successful and unsuccessful attempts, restoring the reward contrast that the natural all-incorrect group lacked. We target this operational *zone of proximal development* with a positive hinter reward. Both $p_{\text{guide}} = 0$ and $p_{\text{guide}} = 1$ yield zero group-relative solver advantages (Eq. (4)) and therefore receive negative hinter feedback. Let $\mathcal{U}_{\text{scored}}$ contain the extraction trajectories whose subgoals were evaluated by guided rollouts, and $\mathcal{U}_{\text{inv}}$ the format-invalid extractions. For reward scale $\alpha > 0$, we use

$$r^{\text{hint}}(u) = \begin{cases} +\alpha, & \text{if } u = u_{q,k} \in \mathcal{U}_{\text{scored}} \text{ and } 0 < p_{\text{guide}}(q,\hat{z}_{q,k}) < 1, \\ -\alpha, & \text{if } u \in \mathcal{U}_{\text{inv}} \text{ or } \big(u = u_{q,k} \in \mathcal{U}_{\text{scored}} \text{ and } p_{\text{guide}}(q,\hat{z}_{q,k}) \in \{0,1\}\big). \end{cases} \tag{6}$$

All mixed outcomes receive the same $+\alpha$, so a subgoal earns full positive credit as soon as it unlocks a learnable group without penalizing $1/G$ frontier breakthroughs (empirical analysis in Section 4.3). For optimization, $\mathcal{B}_{\text{hint}} = \mathcal{U}_{\text{scored}} \cup \hat{\mathcal{U}}_{\text{inv}}$ includes at most $|\mathcal{U}_{\text{scored}}|$ invalid trajectories, sampled uniformly without replacement when necessary. Valid `NONE` abstentions and unevaluated candidates receive no hinter update (Appendix A.1).

3.4 Joint Policy Optimization

Role-specific advantages. Each rollout step produces a solver pool $\mathcal{B}_{\text{solve}}$ of $N_{\text{solve}} = B \times G$ trajectories and a hinter pool $\mathcal{B}_{\text{hint}}$ of N_{hint} extraction trajectories, both sampled from the shared policy $\pi_{\theta_{\text{old}}}$. Solver trajectories use the within-group GRPO advantages in Eq. (1). For hinter trajectory u_m, all response tokens share a batch-centered advantage computed over the full hinter pool, following a REINFORCE++ baseline [Hu, 2025] without standard-deviation normalization:

$$\hat{A}^{\text{hint}}_{m,t} = r^{\text{hint}}(u_m) - \bar{r}^{\text{hint}}, \qquad \bar{r}^{\text{hint}} = \frac{1}{N_{\text{hint}}}\sum_{k=1}^{N_{\text{hint}}} r^{\text{hint}}(u_k). \tag{7}$$

Omitting standard-deviation normalization preserves α as a control on the hinter advantage scale. Setting $\alpha = 0$ removes direct hinter RL while the shared policy continues to learn from solver updates.

Joint update of the shared policy. For each role, we average the clipped token surrogate in Eq. (3) over that pool's active response tokens, using its own advantages, to obtain $\mathcal{J}_{\text{solve}}$ and $\mathcal{J}_{\text{hint}}$. We then maximize their sequence-count-weighted combination:

$$\mathcal{J}_{\text{Joint}}(\theta) = \frac{N_{\text{solve}}}{N_{\text{solve}} + N_{\text{hint}}}\mathcal{J}_{\text{solve}}(\theta) + \frac{N_{\text{hint}}}{N_{\text{solve}} + N_{\text{hint}}}\mathcal{J}_{\text{hint}}(\theta). \tag{8}$$

Normalizing within each pool before combining the objectives prevents longer solver responses from increasing the solver's relative weight solely through their token count. Both objectives update the same parameters θ, providing direct learning signals for solving the original problem and generating useful subgoals. Appendix A.3 gives the expanded objectives and their implementation in a single joint update.

3.5 Why Rescued Backfilling Unlocks all-incorrect Problems

Both FORGE and DAPO's dynamic sampling [Yu et al., 2025] fill the remaining B_{rem} solver slots with mixed-outcome groups ($\sigma_r > 0$), yet direct gradient signal to fundamentally different problems. To isolate why nonzero within-group reward variance alone fails to unlock all-incorrect problems, consider a shared softmax policy π_z over three reasoning modes $m \in \{a, b, c\}$ with probabilities $p_j = e^{z_j} / \sum_u e^{z_u} > 0$: erroneous reasoning ($a$) fails both easy and hard problems, shallow reasoning (b) solves easy problems q_E ($r(q_E, b) = 1$) but fails hard frontier problems q_H ($r(q_H, b) = 0$), and deep reasoning (c) solves both ($r(q_E, c) = r(q_H, c) = 1$, so there is no built-in task conflict). Under dataset weights $\pi_E \in (0, 1)$ and $\pi_H = 1 - \pi_E$, binary verification induces the true dataset expected reward $J(z) = \pi_E(p_b + p_c) + \pi_H p_c$ and unassisted all-incorrect rate $F_G(z) = \pi_E p_a^G + \pi_H(1 - p_c)^G$. At the rollout checkpoint ($\theta = \theta_{\text{old}}$), let $\hat{g}_x$ be the first-step GRPO group gradient for a mixed group of size G with $k \in \{1, \ldots, G-1\}$ successes and step scale $\lambda_k = k(G-k)/[G^2(s_R + \delta)]$, where $s_R = \sqrt{(k/G)(1-k/G)}$; for a subgoal-guided prompt $\tilde{q}_H$, model guidance as an additive logit shift $\tilde{z} = z + \mathbf{b}$ raising $\tilde{p}_c > p_c$.

Theorem 1 (Mode-crowding under natural backfilling vs. single-sample rescue)**.** *In the three-mode shared softmax model above, compare backfilling a discarded all-incorrect slot on q_H with a natural mixed group on q_E (DAPO) against a rescued guided mixed group on $\tilde{q}_H$ (FORGE):*

- *(i)* Natural backfilling crowds out hard modes. *On a mixed group of q_E, the unsampled-winner event ($n_c = 0, n_a > 0, n_b > 0$) yields $\nabla_z p_c^\top \hat{g}_{q_E} = -\lambda_{n_b} p_c(p_b - p_a)$, strictly lowering p_c whenever $p_b > p_a$, and strictly decreasing $J(z)$ whenever $\pi_H p_c(p_b - p_a) > \pi_E p_a(2p_b + p_c)$ (which requires $J > \pi_E$). Moreover, conditioned on any $0 < k < G$, whenever $p_b(p_b - p_c) > p_a(p_b + 2p_c)$, even the expected update shrinks the hard mode: $\nabla_z p_c^\top \mathbb{E}[\hat{g}_{q_E} \mid k] < 0$.*
- *(ii)* Single-sample rescue reverses the update direction. *Once subgoal guidance samples mode c at least once on $\tilde{q}_H$ ($1 \le k \le G-1$), every realized mixed group unconditionally satisfies $\nabla_z p_c^\top \hat{g}_{\tilde{q}_H} \ge \lambda_k p_c(1 - p_c) > 0$. Furthermore, whenever $J \ge \pi_E$—covering every state where natural backfilling can decrease J—every realized group with $k \ge 1$ satisfies $\nabla_z J^\top \hat{g}_{\tilde{q}_H} \ge \lambda_k p_c(1 - J) > 0$.*
- *(iii)* Dataset-wide all-incorrect rate. *Whenever $\pi_H p_c(1 - p_c)^{G-1} \ge \pi_E p_a^G$, every rescued FORGE group ($k \ge 1$) strictly lowers $F_G(z)$ ($\nabla_z F_G^\top \hat{g}_{\tilde{q}_H} < 0$), whereas whenever $p_b(p_b - p_c) > p_a(p_b + 2p_c)$ and G is sufficiently large, natural backfilling on q_E strictly increases $F_G(z)$ both on $n_c = 0$ and in expectation.*

The mechanism is purely geometric. Because binary verification on q_E rewards shallow mode b and deep mode c identically, reinforcing q_E directs most positive credit to z_b when c is scarce, expanding the softmax normalizer faster than e^{z_c} and siphoning probability mass away from c. On an unlocked hard prompt $\tilde{q}_H$, however, binary verification assigns $r = 0$ (negative advantage) to b and $r = 1$ exclusively to c, so sampling c even once ($k = 1$) flips the update direction on both p_c and F_G—providing a theoretical counterpart to our uniform hinter reward for $1/G$ breakthroughs (§3.3). Full proofs of Theorem 1 (i)–(iii), rescue-probability bounds across semantic clusters, and the extension to general parameterized policies π_θ appear in Appendix D.

4 Experiment

4.1 Experiment Setup

Training data. To ensure broad coverage and problem difficulty while maintaining high data efficiency, we curate a compact, two-stage open-source mathematical RL training curriculum (Table 1):

Algorithm 1. FORGE: Reference-Free Subgoals, Semantic Selection, and Joint RL

Require: Question–answer dataset $\mathcal{D}$, solver batch size B, rollout group size G, hinter reward scale $\alpha > 0$, initial policy π_θ.

for each online RL step **do**
  Set $\pi_{\theta_{\text{old}}} \leftarrow \pi_\theta$; initialize natural and guided mixed-group pools $\mathcal{G}^{\text{mix}}_{\text{nat}}, \mathcal{G}^{\text{mix}}_{\text{guide}}$ and hinter pools $\mathcal{U}_{\text{scored}}, \mathcal{U}_{\text{inv}}$ to $\emptyset$.
  Sample B problems $(q, a^\star) \sim \mathcal{D}$ and G natural responses per problem using $\pi_{\theta_{\text{old}}}$.
  Verify responses against $a^\star$ to obtain r^{solve} and $p_{\text{nat}}(q)$; retain natural mixed groups in $\mathcal{G}^{\text{mix}}_{\text{nat}}$.
  for each all-incorrect problem q **in parallel do**
    Sample G hinter trajectories $u_i \sim \pi_{\theta_{\text{old}}}(\cdot \mid \mathcal{P}_{\text{hint}}(q, \text{think}(o_i)))$. ▷ No $a^\star$ or reference solution
    Validate and semantically group subgoals; prepare $\{\hat{z}_{q,k}\}_{k=1}^{K_q}$ with cluster labels and trajectories $u_{q,k}$ (Appendix A.1); collect invalid outputs in $\mathcal{U}_{\text{inv}}$.
    for each candidate $\hat{z}_{q,k}$ **in parallel do**
      Sample G guided responses to $\mathcal{P}_{\text{solve}}(q, \hat{z}_{q,k})$ and verify against $a^\star$ to obtain r^{solve} and $p_{\text{guide}}(q, \hat{z}_{q,k})$.
      Assign $r^{\text{hint}}(u_{q,k}) = +\alpha$ if $0 < p_{\text{guide}} < 1$, else $-\alpha$; add $u_{q,k}$ to $\mathcal{U}_{\text{scored}}$.
      if $0 < p_{\text{guide}} < 1$ **then**
        Add the full guided group to $\mathcal{G}^{\text{mix}}_{\text{guide}}$, retaining its source problem, cluster label, and pass rate.
      end if
    end for
  end for
  Retain all natural mixed groups; fill up to $B - |\mathcal{G}^{\text{mix}}_{\text{nat}}|$ slots with guided mixed groups, covering distinct clusters before repeats and prioritizing lower p_{guide} within each cluster (Section 3.2).
  Fill any residual slots with remaining natural groups to form $\mathcal{B}_{\text{solve}}$ of $B \times G$ trajectories.
  Form $\mathcal{B}_{\text{hint}} = \mathcal{U}_{\text{scored}} \cup \hat{\mathcal{U}}_{\text{inv}}$, sampling at most $|\mathcal{U}_{\text{scored}}|$ invalid outputs and assigning each $-\alpha$ (Section 3.3).
  Compute group-relative solver advantages and batch-centered hinter advantages (Eqs. (1), (7)).
  Jointly update θ using $\mathcal{B}_{\text{solve}}$ and $\mathcal{B}_{\text{hint}}$ under Eq. (8).
end for

- **Stage-1 dataset (16K → 32K context, 14,809 prompts):** We collect candidate problems from six public sources: NemotronRL-math-stack_overflow [NVIDIA, 2025], Big-Math-RL-Verified [Albalak et al., 2025], NemotronRL-math-OpenMathReasoning [NVIDIA, 2025], RLVR-Linearity [Wang et al., 2026], MATH [Hendrycks et al., 2021], and mAceReason-Math [Dobler et al., 2026]. Prior to training, we apply a two-stage curation and difficulty-filtering pipeline alongside exact and semantic deduplication: (1) *Solvability verification*: to filter out noisy or unsolvable items, we query `Qwen3.6-27B` twice per prompt under a 100K-token generation budget and retain only prompts solved at least once; (2) *Difficulty filtering*: we estimate per-prompt pass rates over 8 rollouts from a fixed reference policy under the same 100K-token budget, discarding saturated (8/8) prompts, downsampling 7/8 prompts, and holding out 1,747 frontier (0/8) prompts to balance initial difficulty. The resulting Stage-1 dataset comprises **14,809** verifiable math prompts across the six sources (Table 1, left).

- **Refreshed dataset for 48K and 64K scaling (16,803 prompts):** At step 350, before the response budget extends past 32K, a substantial fraction of Stage-1 problems have become trivial. We refresh the training distribution through three targeted operations (per-source breakdown in Table 1, right): (1) *Online saturation pruning (−**2,653** prompts)*: we inspect the most recent natural rollout group ($G = 16$) of every Stage-1 prompt within the trailing two-epoch training window and filter out all **2,653** prompts that have reached 16/16 accuracy, retaining 12,156 unsaturated Stage-1 problems; (2) *Re-introducing held-out 0/8 frontier problems (+**1,747** prompts)*: we incorporate the **1,747** verified-solvable (0/8) frontier prompts held out during Stage-1 curation; and (3) *Incorporating new frontier sources (+**2,900** prompts)*: we curate **2,900** additional non-trivial (1/8–7/8) prompts from **ArXivMath-Train** [Balunović et al., 2025], **QuestA** [Li et al., 2026], and **NemotronRL-Math-v2** [NVIDIA, 2025] after strict exact/semantic deduplication (cosine < 0.93 against Stage 1) and benchmark decontamination (cosine < 0.60 against all evaluation sets), yielding a final Stage-2 dataset of **16,803** prompts.

Table 1. Source composition and curriculum evolution of the open-source mathematical RL training datasets from Stage 1 (16K → 32K context, 14,809 prompts) to Stage 2 (context scaling, 16,803 prompts), detailing online 16/16 saturation pruning and frontier problem additions.

	Stage 1 (16K → 32K)		Refresh at step 350		48K and 64K	
Source Dataset	**# Prompts**	**Share (%)**	**Dropped (16/16)**	**Added (0/8 / New)**	**# Prompts**	**Share (%)**
NemotronRL-math-stack_overflow [NVIDIA, 2025]	5,148	34.76	−1,140	0	4,008	23.85
Big-Math-RL-Verified [Albalak et al., 2025]	4,243	28.65	−920	0	3,323	19.78
NemotronRL-math-OpenMathReasoning [NVIDIA, 2025]	2,651	17.90	−293	+1,246	3,604	21.45
RLVR-Linearity [Wang et al., 2026]	1,401	9.46	−187	+417	1,631	9.71
MATH [Hendrycks et al., 2021]	959	6.48	−72	+13	900	5.36
mAceReason-Math [Dobler et al., 2026]	407	2.75	−41	+71	437	2.60
ArXivMath-Train [Balunović et al., 2025]	0	0.00	0	+1,099	1,099	6.54
QuestA [Li et al., 2026]	0	0.00	0	+944	944	5.62
NemotronRL-Math-v2 [NVIDIA, 2025]	0	0.00	0	+857	857	5.10
Total	**14,809**	**100.00**	**−2,653**	**+4,647**	**16,803**	**100.00**

Evaluation protocol. We evaluate each model across seven challenging benchmarks (312 problems in total) spanning high-school competitions, Olympiad-level problems, and frontier research mathematics: **AIME25** [MAA, 2025] and **AIME26** [MAA, 2026] (all 30 problems from AIME I and II in each year), **HMMT_Feb25** and **HMMT_Feb26** [Balunović et al., 2025] (30 and 33 problems from the February 2025 and 2026 Harvard–MIT Mathematics Tournaments), **BeyondAIME** [ByteDance Seed Team, 2025] (100 Olympiad-level problems curated to exceed standard AIME difficulty), and **ArXivMath_0526** and **ArXivMath_0626** [Balunović et al., 2025] (40 and 49 uncontaminated research-level mathematical problems extracted from May and June 2026 arXiv preprints, respectively). For each problem, we sample $K = 32$ independent rollouts under a maximum generation budget of 81,920 tokens using a sampling temperature of 1.0, top-$p = 0.95$, and top-$k = 20$, and report both Avg@32 (Pass@1 estimated over 32 rollouts) and Pass@32, together with their problem-weighted **Micro Average** across all 312 problems. All evaluations are conducted using the EvalScope framework [ModelScope Team, 2024] with its built-in symbolic and numeric answer verifiers on extracted `\boxed{}` predictions. Because **ArXivMath_0526** and **ArXivMath_0626** feature diverse symbolic expressions (e.g., functional forms, sets, and intervals) where rule-based string matching can incur false negatives, we augment their rule-based verifier with an LLM-recall step using a generative judge, `Qwen3.6-27B-RM` (configured in thinking mode with a 32,768-token budget), to verify mathematical equivalence on extracted final answers that do not pass the rule-based check (see Appendix B.4).

Baselines and training setup. We compare `FORGE` against the base checkpoint `Qwen3.5-4B-Base`, the official post-trained reasoning model `Qwen3.5-4B`, and four representative RL baselines initialized from `Qwen3.5-4B-Base`: **GRPO** [Shao et al., 2024], **DAPO** [Yu et al., 2025], **HATCH** [Wang et al., 2026], and **SAGE** [Liao et al., 2026]. To ensure a strictly controlled comparison, all RL methods are implemented within the `slime` RL framework [Zhu et al., 2025] under a unified, fully on-policy training setup. At each step, we sample a batch of $B = 128$ prompts with $G = 16$ rollouts per prompt (yielding a matched natural batch of 2,048 trajectories per gradient update across all methods, where guided groups in SAGE and `FORGE` only backfill zero-variance natural slots, while HATCH augments the original batch with hinted solver and hinter branches on all-incorrect prompts) using a sampling temperature of 1.0 and default top-p and top-k settings. Across all methods, we optimize the token-level policy gradient loss in Eq. (3) with asymmetric clipping ($\epsilon_{\text{low}} = 0.2$, $\epsilon_{\text{high}} = 0.28$), without KL or entropy regularization ($\beta = 0$, entropy coefficient 0.0), and without language or format penalties. During the matched comparison, models are trained using the Adam optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.98$, weight decay 0.1) with a cosine learning rate schedule decaying from 1×10^{-6} to 1×10^{-7}. Following the progressive context-expansion curriculum [Luo et al., 2025], all methods are first trained under a 16,384-token maximum response length for 220 steps and then under a 32,768-token budget for an additional 130 steps, constituting the 350-step fair comparison (Stage 1). To further explore the long-horizon scaling potential of `FORGE`, we continue training it in Stage 2 for 100 steps under a 49,152-token budget (steps 350–449) and 50 steps under a 65,536-token budget (steps 450–499). Implementation details for `FORGE` are provided in Appendix A. Baseline-specific configurations are as follows:

Table 2. Main mathematical reasoning results across seven competition, Olympiad, and research-level benchmarks (312 problems in total) under an 81,920-token generation budget ($K = 32$ rollouts per problem). We report Avg@32 (Pass@1) and Pass@32 in percent. **Micro Average** weights each of the 312 problems equally. Best results in each metric column are highlighted in **bold**, and the best controlled Stage-1 results (350 steps) are underlined. Rows of FORGE are shaded.

Model / Method	AIME 25		AIME 26		HMMT Feb25		HMMT Feb26		Beyond AIME		ArXivMath 0526		ArXivMath 0626		Micro Average	
	Avg@32	Pass@32	Avg@32	Pass@32	Avg@32	Pass@32	Avg@32	Pass@32	Avg@32	Pass@32	Avg@32	Pass@32	Avg@32	Pass@32	Avg@32	Pass@32
Reference Models																
`Qwen3.5-4B-Base`	81.35	**96.67**	82.50	**96.67**	74.27	**100.00**	68.47	87.88	56.66	85.00	24.84	62.50	10.91	55.10	53.20	81.41
`Qwen3.5-4B`	85.00	90.00	84.17	93.33	76.46	90.00	62.22	84.85	56.31	74.00	24.06	50.00	12.63	36.73	53.32	71.15
Stage 1: Controlled RL Comparison on `Qwen3.5-4B-Base` (350 Steps, 16K → 32K Context)																
GRPO	56.25	86.67	64.48	86.67	41.15	73.33	41.10	84.85	39.66	67.00	18.75	47.50	9.25	48.98	36.48	67.95
DAPO	54.79	86.67	67.08	90.00	42.29	70.00	46.12	81.82	40.59	68.00	20.31	50.00	8.74	36.73	37.65	66.35
HATCH	58.54	83.33	66.88	90.00	38.44	70.00	46.12	78.79	40.75	71.00	20.63	47.50	9.25	42.86	37.79	67.31
SAGE	79.06	93.33	82.60	**96.67**	66.04	86.67	60.61	**90.91**	51.44	78.00	27.34	67.50	13.33	48.98	50.39	77.56
FORGE	80.31	**96.67**	82.71	**96.67**	65.52	93.33	62.88	87.88	53.38	84.00	28.75	**70.00**	13.90	**63.27**	51.60	82.69
Stage 2: Long-Horizon Context Scaling on `Qwen3.5-4B-Base`																
FORGE	**86.77**	**96.67**	**89.48**	**96.67**	**78.33**	93.33	**69.70**	**90.91**	**59.59**	**87.00**	**34.92**	67.50	**18.75**	61.22	**58.37**	**83.33**

- **GRPO** [Shao et al., 2024]: We sample exactly 128 prompts per step without dynamic filtering or oversampling, retaining both all-correct and all-incorrect rollout groups (which receive zero advantage) as well as truncated trajectories.
- **DAPO** [Yu et al., 2025]: We enable dynamic sampling to filter out zero-variance groups (all-correct or all-incorrect) using a 3× oversampling pool of 384 prompts per step to backfill discarded groups, while disabling the overlong reward penalty.
- **HATCH** [Wang et al., 2026]: Using the same verified reference solutions as SAGE, for each all-incorrect (0/16) natural group, the shared policy generates 4 concise reference-conditioned hints (non-thinking mode, 1,024-token budget, temperature 0.3) and $G = 16$ solver rollouts per valid hint, jointly updating hinter and solver with hint-advantage annealing ($\gamma = 5$) and conflict-averse gradient projection [Wang et al., 2026]. Because HATCH leaves the hint reward functional unspecified, we instantiate it with the transfer-weighted non-degenerate probability reward of HiLL [Xia et al., 2026], using Eq. (12) ($T = 0.3$, detailed in Appendix C.2).
- **SAGE** [Liao et al., 2026]: For each training prompt, we pair the problem with a verified reference solution trajectory collected during the *solvability verification* stage. Following the official implementation and hyperparameter settings [Liao et al., 2026], when an initial unhinted probe yields an all-incorrect (0/16) group, the policy iteratively generates and conditions on up to $L = 3$ levels of privileged self-hints (with a 4,096-token hint budget under the non-thinking mode, temperature 0.6, and up to 3 generation retries) to recover positive solving trajectories.

4.2 Main Results

Starting from `Qwen3.5-4B-Base` and using within 20K open-source question–answer problems across training, FORGE exceeds the official post-trained model within 500 steps (Table 2). At the matched 350-step checkpoint, it already leads the controlled RL comparison without reference solutions.

Stronger controlled RL results without reference solutions. At the matched checkpoint, under the same training data and context schedule, FORGE achieves the highest aggregate accuracy and problem coverage among the RL methods. Its micro-average Avg@32 exceeds GRPO, DAPO, and HATCH by **15.12**, **13.95**, and **13.81 percentage points**, respectively. Notably, although DAPO expends up to a 3× unassisted rollout

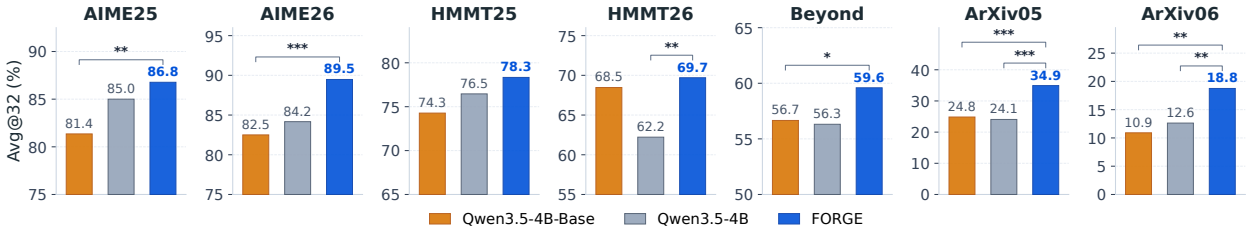

Figure 5. Accuracy gains across benchmarks after context scaling. Bars show Avg@32 for FORGE and the two reference models under the evaluation protocol of Table 2; each panel uses a benchmark-specific vertical scale with exact values labeled above each bar. Significance is assessed by two-sided paired permutation tests on problem-level accuracies, averaging the 32 responses within each problem. Brackets identify significant comparisons using unadjusted p-values: $^{*}p < 0.05$, $^{**}p < 0.01$, and $^{***}p < 0.001$.

budget per step ($384 \times 16 = 6{,}144$ trajectories) to dynamically resample past zero-variance groups, it improves Avg@32 over GRPO by only 1.17 points while reducing Pass@32 by 1.60 points (66.35% vs. 67.95%). Consistent with prior evidence that extra unguided rollouts on failed problems yield diminishing returns [Tang et al., 2026], brute-force unassisted resampling concentrates updates on already solvable problems rather than breaking through reasoning impasses on all-incorrect problems. Even with privileged reference solutions and joint hinter–solver updates, HATCH reaches only 37.79% Avg@32 and 67.31% Pass@32—barely ahead of GRPO (+1.31 points in Avg@32, −0.64 points in Pass@32) and essentially tied with DAPO. In practice, HATCH's fifth-power annealing schedule suppresses hinted solver advantages to near zero once natural mixed groups outnumber all-incorrect groups ($\beta \approx 1.2 \times 10^{-3}$ in the 32K stage), while length-normalized transfer rewards distort hinter credit on long reasoning chains (Appendix C.2), effectively collapsing its solver update toward standard GRPO. Compared with SAGE, the strongest reference-guided baseline, FORGE improves Avg@32 by **1.21 points** and Pass@32 by **5.13 points**, solving 21 benchmark problems where SAGE scores 0/32 while losing only 5 (Table 14). The larger gain in Pass@32 reflects broader multi-attempt problem coverage, showing that guidance derived from the policy's own failures outperforms reference-solution supervision.

Surpassing the official model with further context scaling. After further training steps under a larger response budget, FORGE reaches **58.37%** micro-average Avg@32 and **83.33%** Pass@32. These results exceed the official `Qwen3.5-4B` by **5.05** and **12.18 percentage points**, respectively, and improve both metrics over the base checkpoint. These simultaneous gains show that FORGE improves single-sample accuracy while expanding multi-attempt problem coverage. The accuracy gains span benchmarks with widely varying baseline performance, from competition mathematics to the more challenging research-level ArXivMath problems (Figure 5). FORGE improves Avg@32 over both reference models on all seven benchmarks; on six of them, the improvement is statistically significant against at least one reference model under a two-sided paired permutation test (unadjusted $p < 0.05$). On both ArXivMath benchmarks, the gains are significant against both references, showing that the improvement also extends to the low-accuracy regime.

4.3 Hinter RL Ablation and Reward Design

We examine the role of direct hinter optimization and our uniform hinter reward design at the controlled 350-step checkpoint (16K → 32K schedule). Figure 6 compares the default reward scale $\alpha = 0.3$ against a reduced scale $\alpha = 0.1$ and solver-only updates without direct hinter RL ($\alpha = 0$), alongside our pass-rate-stratified subgoal quality audit (complete per-benchmark scores appear in Table 11 in Appendix C).

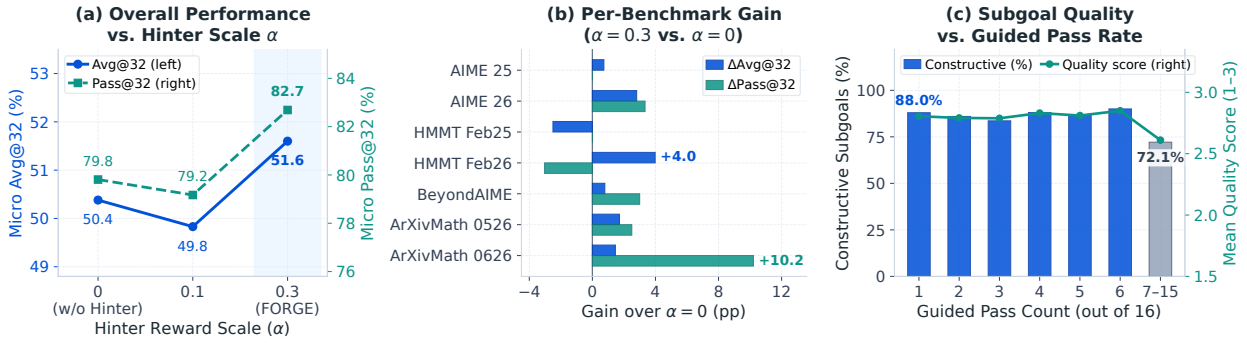

Figure 6. Direct hinter RL requires sufficient reward scale, and uniform credit preserves subgoal quality without distortion. (a) Overall performance vs. hinter scale α: Under an identical 350-step budget, the default $\alpha = 0.3$ (FORGE) achieves the highest micro-average Avg@32 and Pass@32 across all seven math benchmarks, whereas $\alpha = 0.1$ is too weak to improve over solver-only updates ($\alpha = 0$). **(b) Per-benchmark gain ($\alpha = 0.3$ vs. $\alpha = 0$):** Direct hinter RL raises Avg@32 on six of seven benchmarks and delivers the largest Pass@32 improvements on the hardest Olympiad and research-level suites (up to +10.2 pp on ArXivMath 0626). **(c) Subgoal quality vs. guided pass rate:** Frontier 1/16 breakthroughs match moderate pass-rate groups (2–6/16) in both constructive share (bars) and quality score (teal) while outperforming high pass-rate groups (7–15/16), supporting uniform credit over midpoint-peaking shaping.

Hinter reward scale matters for downstream performance. Because all three settings execute the same subgoal-extraction and guided-rollout pipeline per all-incorrect problem, they disentangle the value of learned subgoal quality from the auxiliary rollout budget itself. The default $\alpha = 0.3$ improves micro-average Avg@32 and Pass@32 over $\alpha = 0$ by $+\mathbf{1.22}$ and $+\mathbf{2.88}$ percentage points, with the strongest coverage gains concentrated on the hardest benchmarks (Figure 6a–b). Because hinter advantages are mean-centered without standard-deviation normalization (Eq. (7)), α directly controls the magnitude of the hinter update: an overly small weight ($\alpha = 0.1$) provides too weak a gradient to improve over $\alpha = 0$, whereas $\alpha = 0.3$ supplies a sufficient learning signal to elevate both accuracy and exploration coverage.

Direct hinter RL sustains constructive subgoals. Solver updates reward solving the original problem but provide no direct training signal for formulating intermediate subproblems. As shown in Figure 3a, removing direct hinter RL ($\alpha = 0$) leaves a persistent quality gap despite continued updates to the shared policy, whereas joint optimization in FORGE sustains high constructive guidance throughout training and reaches **90.4**% (category breakdown in Figure 10; qualitative examples in Appendix E.3).

Uniform credit avoids distorting subgoal quality. In outcome-based RLVR, GRPO gradient variance peaks near a 50% pass rate, which might naturally suggest weighting hinter rewards toward mid-range guided pass rates. However, p_{guide} reflects the remaining difficulty of completing the *original problem* rather than the intrinsic quality of the proposed subgoal. Our blind stratified audit confirms that subgoal quality is independent of—and slightly negatively correlated with—guided pass rate (Figure 6c; Tables 9–10): 1/16 breakthroughs achieve **88.0**% constructive subgoals, matching moderate pass-rate groups (2–6/16) and outperforming high pass-rate groups (7–15/16), which contain more restatements. Assigning a simple uniform $+\alpha$ to every mixed group therefore avoids down-weighting genuine frontier breakthroughs without introducing reward distortion (case studies in Appendix E.4).

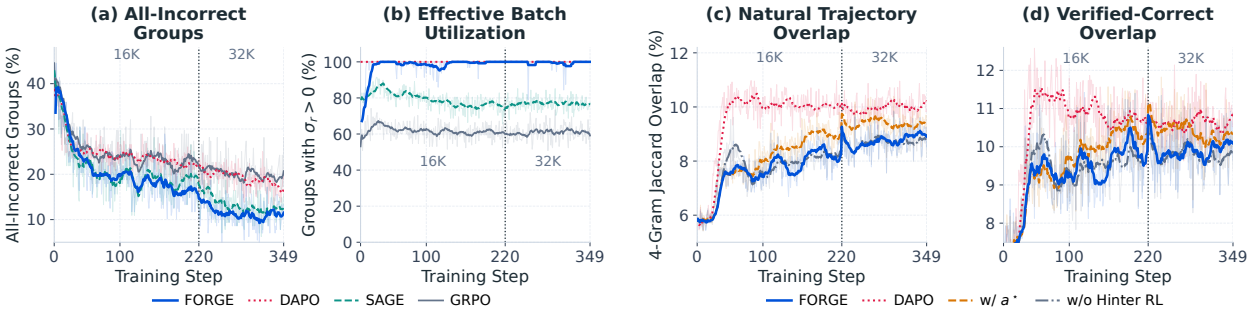

Figure 7. Training dynamics and natural trajectory overlap across 350 RL steps. The rollout budget increases from 16K to 32K at step 220 (vertical dotted lines). Faint curves show per-step measurements; bold curves show 12-step trailing means, restarted at the budget change in (c,d). **(a)** Fraction of natural groups that are all-incorrect (0/16). **(b)** Fraction of the 128-group solver batch with non-zero reward variance ($\sigma_r > 0$). **(c,d)** BPE 4-gram Jaccard overlap among all $G = 16$ unassisted rollouts per problem and among verified-correct rollouts, respectively; (d) includes only problems with at least four correct responses. Overlap is measured on the 128 incoming natural prompts with rollouts rank-matched by token length up to 16K tokens; lower values indicate greater lexical diversity.

4.4 Training Dynamics and Sample Efficiency

Figure 7 brings together training effectiveness and natural trajectory overlap across the 350-step training horizon. Panels (a,b) track the remaining all-incorrect groups and effective batch utilization; panels (c,d) examine lexical diversity in unassisted rollouts. Together, they show that FORGE combines fewer all-incorrect groups and near-full effective batches with lower trajectory overlap than DAPO.

Accelerated mastery of all-incorrect frontier problems (Figure 7a). At the onset of RL training under the 16K budget, roughly 38%–42% of natural prompt groups fail completely (0/16). Under GRPO and DAPO, the natural all-incorrect ratio plateaus at an average of 20.14% and 19.00% during the 32K stage (steps 220–349), indicating that nearly one-fifth of the dataset remains beyond the policy's reach (similarly 20.54% under HATCH, whose rapid annealing curtails hinted solver updates). Even though DAPO draws a 3× oversampled pool of 384 prompts per step to discard zero-variance groups, simply skipping all-incorrect problems never teaches the policy how to solve them. In contrast, FORGE directly targets these 0/16 groups with self-generated subgoal hints, driving the natural all-incorrect ratio down to an average of **11.33**% in the 32K stage—the lowest among all methods and a **40**%–**44**% relative reduction in unsolvable frontier groups compared to DAPO and GRPO.

Near-100% effective batch utilization without prompt oversampling (Figure 7b). In standard GRPO, retaining all-correct and all-incorrect groups leaves only 60.83% of the 128-group batch with non-zero reward variance ($\sigma_r > 0$) during steps 220–349, wasting nearly 40% of actor forward–backward compute on zero-gradient sequences. While SAGE recovers part of these lost slots via progressive reference-guided hints, its single-path escalation still leaves batch utilization at 76.89%. By evaluating multiple semantic subgoal clusters per failed problem and selecting unlocked groups via semantic-cluster layer round-robin, FORGE fills **99.53**% of the 128 solver batch slots with informative mixed groups ($\sigma_r > 0$), alongside DAPO's 100% batch utilization while sampling only 128 natural prompts per step instead of DAPO's 384 prompts (rollout and update GPU-hour comparisons appear in Appendix C.3).

Preserving exploration through diverse subgoal guidance (Figure 7c,d). FORGE combines near-full effective batch utilization with lower trajectory overlap than DAPO, reducing average overlap by **17.3%** across

unassisted rollouts and **9.5%** among verified-correct responses. Whereas DAPO replenishes the batch by drawing additional solvable problems, FORGE revisits all-incorrect problems through distinct semantic subgoals, countering within-problem homogenization through alternative reasoning entry points.

Exposing the ground-truth answer to the hinter increases overlap across both rollout populations through *target-anchored homogenization*: a shared target steers extractions from different failed attempts toward similar backward subgoals. Withholding the answer instead directs the hinter to extract distinct intermediate quantities from the failed attempts themselves.

Removing direct hinter RL leaves trajectory overlap largely unchanged but lowers downstream accuracy and coverage. This reflects a clear division of roles within FORGE: reference-free extraction and semantic clustering preserve diverse reasoning entry points, while hinter optimization sharpens their utility for solving. Joint training thus makes better use of the retained diversity.

4.5 High-Budget Exploration Scaling

A central question in reinforcement learning for mathematical reasoning is whether sharpening single-sample accuracy narrows the policy's exploration boundary under large test-time sampling budgets [Yue et al., 2025, Cui et al., 2025, Liu et al., 2025]. To evaluate whether the trajectory diversity preserved during training translates into sustained test-time exploration, we compare FORGE against `Qwen3.5-4B-Base` when scaling the rollout horizon up to $K = 512$.

High-budget evaluation protocol. Starting from the $K = 32$ rollouts collected under the standard evaluation protocol of Section 4.1, we progressively double the sampling budget across four additional stages: $k \in \{64, 128, 256, 512\}$. Because any problem solved at least once within a given stage is guaranteed to remain solved at all larger budgets, we allocate additional rollouts at each stage exclusively to the subset of problems that remain completely unsolved (0 correct) at the preceding horizon, and apply the same rule-based and generative-judge verification pipeline (including `Qwen3.6-27B-RM` on **ArXivMath_0526** and **ArXivMath_0626**). This progressive schedule yields exact problem-level coverage checkpoints at $k \in \{32, 64, 128, 256, 512\}$ together with continuous Pass@k curves from $k = 1$ to $k = 512$ across all seven benchmarks (312 problems in total).

Preserving exploration breadth from $k = 1$ to $k = 512$ (Figure 8a). Conventional post-training often trades multi-sample coverage for single-sample accuracy [Yue et al., 2025, Cui et al., 2025]: although the official `Qwen3.5-4B` model matches `Qwen3.5-4B-Base` at $k = 1$, it falls behind by 10.26 percentage points at Pass@32. In contrast, FORGE improves single-sample accuracy by $+$**5.17** percentage points over the base model while keeping its micro-average Pass@k curve above or tied with the base at every sampling horizon from $k = 1$ to $k = 512$, without crossover. Its advantage thus extends well beyond the standard $k = 32$ evaluation budget: as sampling expands a further $16\times$, FORGE maintains a positive scaling slope and reaches **91.99**% coverage across all 312 problems, compared with 90.71% for the base model (Table 15).

Sustained gains on unsaturated Olympiad and research-level benchmarks (Figure 8b–e). The exploration advantage of FORGE is clearest on benchmarks that still demand deeper search after the standard evaluation budget is exhausted. While **AIME25** and **AIME26** already saturate at 29/30 for both models by $k = 32$, FORGE consistently leads the base model throughout the extended sampling regime on the four unsaturated Olympiad and research-level benchmarks, which account for 222 of the 312 evaluation problems. This sustained lead culminates in **90.09**% pooled coverage at $k = 512$, solving **200**/**222** problems versus 195/222

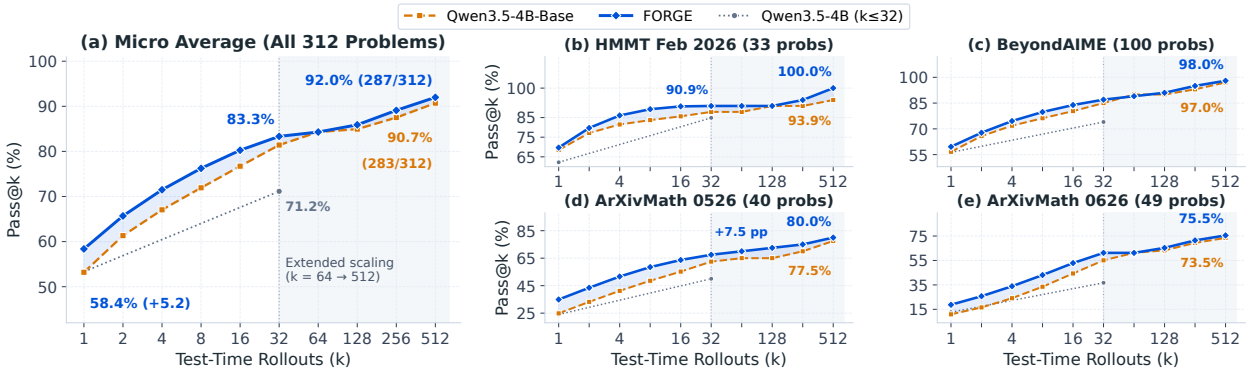

Figure 8. Test-time Pass@k exploration scaling from $k = 1$ to $k = 512$. We compare `FORGE` against `Qwen3.5-4B-Base` across $k \in \{1, 2, 4, 8, 16, 32, 64, 128, 256, 512\}$ rollouts per problem under an 81,920-token generation budget, alongside the official `Qwen3.5-4B` reference at $k \leq 32$. The vertical dotted line at $k = 32$ separates the standard evaluation budget from the extended high-budget exploration regime ($k \in [64, 512]$, shaded). **(a)** Problem-weighted micro average across all 312 benchmark problems. **(b–e)** Scaling curves on the four unsaturated Olympiad and frontier research-level benchmarks (222 problems in total). Complete numerical results for all seven benchmarks appear in Table 15.

Table 3. Science and STEM multiple-choice accuracy (%).

Model / Method	GPQA-Diamond	SuperGPQA-STEM-Hard	HLE-Science-MC	MMLU-Pro-STEM	Average
`Qwen3.5-4B-Base`	62.83	43.90	10.88	82.08	49.92
DAPO	48.81	84.94	6.81	68.38	52.24
GRPO	48.08	84.34	9.05	65.56	51.76
SAGE	48.97	84.36	6.73	68.38	52.09
HATCH	–	–	–	–	–
`FORGE`	59.02	85.40	11.46	70.61	54.12

for the base model. On the Olympiad benchmarks, `FORGE` achieves complete coverage of **HMMT_Feb26** (**33**/**33** solved, versus 31/33 for the base model) and near-complete coverage of **BeyondAIME** (**98**/**100**). The research-level benchmarks show substantial room for continued exploration: increasing the budget from $k = 32$ to $k = 512$ unlocks an additional $+$**12.50** and $+$**14.29** percentage points of coverage on **ArXivMath_0526** and **ArXivMath_0626**, respectively. Here, `FORGE` at $k = 256$ already matches or approaches the base model's coverage at $k = 512$, reaching comparable coverage with half the sampling budget. Unlocking all-incorrect problems during training through diverse, reference-free subgoals thus elevates point accuracy while preserving—and on hard benchmarks expanding—the base model's exploration frontier.

4.6 Transfer to Science and STEM Reasoning

To validate the generalizability of `FORGE`, we compare it to other RL-based methods on science and STEM reasoning. We curate a training subset from the Nemotron-RL-knowledge-mcqa dataset [1] and evaluate each method on GPQA-Diamond, SuperGPQA-STEM-Hard, HLE-Science-MC, and MMLU-Pro-STEM (Details in Appendix A.5). Results in Table 3.

[1] https://huggingface.co/datasets/nvidia/Nemotron-RL-knowledge-mcqa

Table 4. Comparison of FORGE against representative RLVR and hint-guided reasoning frameworks across six key design dimensions.

Method	Unlocks all-incorrect Frontier	No Reference Solution or Answer in Guidance	Multi-Entry Guidance	Strictly On-Policy	Single-Policy Co-Evolution	Zero Test-Time Overhead
GRPO [Shao et al., 2024]	✗	✓	✗	✓	—	✓
DAPO [Yu et al., 2025]	✗	✓	✗	✓	—	✓
MINER [Jiang et al., 2026]						
ExGRPO [Zhan et al., 2025]	✗	✓	✗	✗	—	✓
R^3 [Jiang et al., 2026]						
CURE [Chen et al., 2026]	✓	✓	✗	✓	✓	✗
A^2D [Chen et al., 2026]	✓	✓	✗	✓	✗	✓
LTE [Tang et al., 2026]	✓	✓	✗	✗	—	✓
RLTL;DR [Kirchhof et al., 2026]	✓	✓	✗	✓	✓	✓
QuestA [Li et al., 2026], GHPO [Liu et al., 2025], BHA [Xie et al., 2026]						
Guide-GRPO [Nath et al., 2025], NuRL [Chen et al., 2026]	✓	✗	✗	✓	✗	✓
SAGE [Liao et al., 2026], HDPO [Ding, 2026]						
HiLL [Xia et al., 2026]	✓	✗	✓	✓	✗	✓
HATCH [Wang et al., 2026]	✓	✗	✓	✓	✓	✓
FORGE **(Ours)**	✓	✓	✓	✓	✓	✓

5 Related Work

We situate FORGE within research on learning from uniform-outcome groups in group-relative RL and on using intermediate guidance to improve reasoning. Table 4 summarizes the main design differences.

Learning Signals in Group-Relative Optimization Recent reasoning models [Jaech et al., 2024, Guo et al., 2025, Kimi Team et al., 2025, Yang et al., 2024, 2025] scale RLVR [Hu et al., 2025, Zeng et al., 2025, Yue et al., 2025] with critic-free policy gradients [Schulman et al., 2017, Ahmadian et al., 2024, Shao et al., 2024, Hu, 2025]. GRPO [Shao et al., 2024] yields no group-relative signal when all outcomes coincide. DAPO [Yu et al., 2025] filters such groups and resamples, while MINER [Jiang et al., 2026] continues learning from all-correct groups via uncertainty-based credit assignment. Both leave open how to elicit successful attempts on all-incorrect problems, and concentrating updates on already solvable cases can narrow policy entropy and multi-sample coverage [Yue et al., 2025, Cui et al., 2025, Liu et al., 2025]. ExGRPO [Zhan et al., 2025] and R^3 [Jiang et al., 2026] recover contrast via historical trajectories, but introduce off-policy data and cannot help problems with no recorded success. FORGE instead constructs intermediate subgoals from the current policy's failed attempts and revisits the original problem on-policy.

Hint-Based Exploration and Problem Decomposition Decomposing reasoning into intermediate steps has long supported chain-of-thought prompting [Wei et al., 2022, Zhou et al., 2023], self-taught bootstrapping [Zelikman et al., 2022], and step-level verification [Lightman et al., 2024]. In RLVR, privileged guidance recovers signal on hard problems through partial reference prefixes [Li et al., 2026, Liu et al., 2025, Xie et al., 2026], answer- or reference-conditioned hints [Nath et al., 2025, Chen et al., 2026, Liao et al., 2026, Xia et al., 2026, Wang et al., 2026], or privileged self-distillation [Ding, 2026]. These methods differ in hinter training and guidance annealing, but all condition guidance on reference solutions or ground-truth answers; related designs freeze a question–answer decomposer [Chen et al., 2026] or retain multi-turn critique at test time [Chen et al., 2026]. Without external solutions, LTE [Tang et al., 2026] prompts retries with a rule-based list of prior incorrect final answers under mixed-policy updates, and RLTL;DR [Kirchhof et al., 2026] sequentially accumulates one-sentence verifier-error summaries across retries while supervising the in-context insight tokens. FORGE instead turns intermediate bottlenecks inside failed reasoning prefixes into parallel, semantically clustered question-only subgoals, jointly updates hinter and solver in a single on-policy RL loop under standard outcome verification, and solves unassisted at test time.

6 Conclusion

We identified a recurring pattern in all-incorrect mathematical reasoning problems: failed attempts often make meaningful progress before reaching recoverable reasoning impasses. Motivated by this finding, we proposed `FORGE`, which combines reference-free subgoal generation, semantic selection, and a reward for unlocking all-incorrect problems, with the hinter and solver jointly trained in a single policy. Across seven mathematical benchmarks, `FORGE` improves micro-average Pass@1 over GRPO by 15.12 percentage points under matched training conditions. With further context scaling, training from `Qwen3.5-4B-Base` within 20K open-source problems and 500 steps surpasses the official `Qwen3.5-4B` by 5.05 percentage points in Pass@1 and 12.18 percentage points in Pass@32.

References

[1] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. OpenAI o1 system card. *arXiv preprint arXiv:2412.16720*, 2024. URL https://arxiv.org/abs/2412.16720.

[2] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. URL https://arxiv.org/abs/2501.12948.

[3] Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1.5: Scaling reinforcement learning with LLMs. *arXiv preprint arXiv:2501.12599*, 2025. URL https://arxiv.org/abs/2501.12599.

[4] Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-Reasoner-Zero: An open source approach to scaling up reinforcement learning on the base model. *arXiv preprint arXiv:2503.24290*, 2025. URL https://arxiv.org/abs/2503.24290.

[5] Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. SimpleRL-Zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*, 2025. URL https://arxiv.org/abs/2503.18892.

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

[7] Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12248–12267, 2024. URL https://aclanthology.org/2024.acl-long.662/.

[8] Jian Hu. REINFORCE++: A simple and efficient approach for aligning large language models. *arXiv preprint arXiv:2501.03262*, 2025. URL https://arxiv.org/abs/2501.03262.

[9] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025. URL https://arxiv.org/abs/2503.14476.

[10] Runzhe Zhan, Yafu Li, Zhi Wang, Xiaoye Qu, Dongrui Liu, Jing Shao, Derek F. Wong, and Yu Cheng. ExGRPO: Learning to reason from experience. *arXiv preprint arXiv:2510.02245*, 2025. URL https://arxiv.org/abs/2510.02245.

[11] Zhizheng Jiang, Kang Zhao, Weikai Xu, Xinkui Lin, Wei Liu, Jian Luan, Shuo Shang, and Peng Han. R^3: Replay, reflection, and ranking rewards for LLM reinforcement learning. *arXiv preprint arXiv:2601.19620*, 2026. URL https://arxiv.org/abs/2601.19620.

[12] Jiazheng Li, Hongzhou Lin, Hong Lu, Kaiyue Wen, Zaiwen Yang, Jiaxuan Gao, Yi Wu, and Jingzhao Zhang. QuestA: Expanding reasoning capacity in LLMs via question augmentation. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2507.13266.

[13] Ziru Liu, Cheng Gong, Xinyu Fu, Yaofang Liu, Ran Chen, Shoubo Hu, Suiyun Zhang, Rui Liu, Qingfu Zhang, and Dandan Tu. GHPO: Adaptive guidance for stable and efficient LLM reinforcement learning. *arXiv preprint arXiv:2507.10628*, 2025. URL https://arxiv.org/abs/2507.10628.

[14] Pei-Xi Xie, Che-Yu Lin, and Cheng-Lin Yang. Mitigating distribution sharpening in math RLVR via distribution-aligned hint synthesis and backward hint annealing. *arXiv preprint arXiv:2604.07747*, 2026. URL https://arxiv.org/abs/2604.07747.

[15] Vaskar Nath, Elaine Lau, Anisha Gunjal, Manasi Sharma, Nikhil Baharte, and Sean Hendryx. Adaptive guidance accelerates reinforcement learning of reasoning models. *arXiv preprint arXiv:2506.13923*, 2025. URL https://arxiv.org/abs/2506.13923.

[16] Justin Chih-Yao Chen, Becky Xiangyu Peng, Prafulla Kumar Choubey, Kung-Hsiang Huang, Jiaxin Zhang, Mohit Bansal, and Chien-Sheng Wu. Nudging the boundaries of LLM reasoning. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2509.25666.

[17] Ken Ding. HDPO: Hybrid distillation policy optimization via privileged self-distillation. *arXiv preprint arXiv:2603.23871*, 2026. URL https://arxiv.org/abs/2603.23871.

[18] Yu Xia, Canwen Xu, Zhewei Yao, Julian McAuley, and Yuxiong He. Learning to hint for reinforcement learning. *arXiv preprint arXiv:2604.00698*, 2026. URL https://arxiv.org/abs/2604.00698.

[19] Zile Wang, Zijian Li, Haodong Wang, Jian Liu, Qianli Liu, Lucas Muli, Blaze Chen, and Song Guo. Teach to learn: Hint annealing for self-improving LLM reasoning. *arXiv preprint arXiv:2609.34975*, 2026. URL https://arxiv.org/abs/2609.34975.

[20] Baohao Liao, Hanze Dong, Xinxing Xu, Christof Monz, and Jiang Bian. Self-hinting language models enhance reinforcement learning. *arXiv preprint arXiv:2602.03143*, 2026. URL https://arxiv.org/abs/2602.03143.

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

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

[23] Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTi Fan, Zhengyin Du, et al. VAPO: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*, 2025. URL https://arxiv.org/abs/2504.05118.

[24] NVIDIA. Nemo gym: An open source library for scaling reinforcement learning environments for llm. https://github.com/NVIDIA-NeMo/Gym, 2025. GitHub repository.

[25] Alon Albalak, Duy Phung, Nathan Lile, Rafael Rafailov, Kanishk Gandhi, Louis Castricato, Anikait Singh, Chase Blagden, Violet Xiang, Dakota Mahan, and Nick Haber. Big-math: A large-scale, high-quality math dataset for reinforcement learning in language models, 2025. URL https://arxiv.org/abs/2502.17387.

[26] Tianle Wang, Jiayu Liu, Zhongyuan Wu, Shenghao Jin, Wei Chen, Hao Xu, and Ning Miao. Linear dynamics in the rlvr training of large language models, 2026. URL https://arxiv.org/abs/2601.04537.

[27] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems (NeurIPS) Track on Datasets and Benchmarks*, 2021. URL https://arxiv.org/abs/2103.03874.

[28] Konstantin Dobler, Simon Lehnerer, Federico Scozzafava, Jonathan Janke, and Mohamed Ali. macereason-math: A dataset of high-quality multilingual math problems ready for rlvr, 2026. URL https://arxiv.org/abs/2603.10767.

[29] Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. MathArena: Evaluating LLMs on uncontaminated math competitions. *arXiv preprint arXiv:2505.23281*, 2025. URL https://arxiv.org/abs/2505.23281.

[30] Mathematical Association of America. 2025 American Invitational Mathematics Examination (AIME I and II). AIME I and AIME II, 2025.

[31] Mathematical Association of America. 2026 American Invitational Mathematics Examination (AIME I and II). AIME I and AIME II, 2026.

[32] ByteDance Seed Team. Seed1.5-Thinking: Advancing superb reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.13914*, 2025. URL https://huggingface.co/datasets/ByteDance-Seed/BeyondAIME.

[33] ModelScope Team. EvalScope: Evaluation framework for large models. https://github.com/modelscope/evalscope, 2024.

[34] Zilin Zhu, Chengxing Xie, Xin Lv, and slime Contributors. slime: An LLM post-training framework for RL scaling. https://github.com/THUDM/slime, 2025. GitHub repository.

[35] Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2, 2025. Notion Blog.

[36] Chenming Tang, Hsiu-Yuan Huang, Weijie Liu, Clive Bai, Saiyong Yang, and Yunfang Wu. Do not step into the same river twice: Learning to reason from trial and error. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8018–8032, 2026. URL https://aclanthology.org/2026.acl-long.363/.

[37] Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025. URL https://arxiv.org/abs/2504.13837.

[38] Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, et al. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025. URL https://arxiv.org/abs/2505.22617.

[39] Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Pavlo Molchanov. ProRL: Prolonged reinforcement learning expands reasoning boundaries in large language models. *arXiv preprint arXiv:2505.24864*, 2025. URL https://arxiv.org/abs/2505.24864.

[40] Shuyang Jiang, Yuhao Wang, Ya Zhang, Yanfeng Wang, and Yu Wang. MINER: Mining intrinsic mastery for data-efficient RL in large reasoning models. *arXiv preprint arXiv:2601.04731*, 2026. URL https://arxiv.org/abs/2601.04731.

[41] Guirong Chen, Shuqi Ye, Wenkai Yang, Shiqi Shen, Guangyao Shen, and Yankai Lin. CURE: Critique-driven unified reinforcement learning for test-time self-improvement. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28632–28653, 2026. URL https://aclanthology.org/2026.acl-long.1321/.

[42] Zhipeng Chen, Xiaobo Qin, Wayne Xin Zhao, Youbin Wu, and Ji-Rong Wen. Adaptive ability decomposing for unlocking large reasoning model effective reinforcement learning. *arXiv preprint arXiv:2602.00759*, 2026. URL https://arxiv.org/abs/2602.00759.

[43] Michael Kirchhof, Eleonora Gualdoni, Andrew Szot, Khashayar Gatmiry, Aryo Lotfi, Abbas Kazerouni, Omar Attia, Sanjoy Chowdhury, and Alexander Toshev. RLTL;DR: Self-improvement by internalizing self-generated feedback. *arXiv preprint arXiv:2609.37633*, 2026. URL https://arxiv.org/abs/2609.37633.

[44] An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2.5-Math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024. URL https://arxiv.org/abs/2409.12122.

[45] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

[46] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837, 2022.

[47] Denny Zhou, Nathanael Schäarli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WZH7099tgfM.

[48] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, volume 35, pages 15476–15488, 2022.

[49] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

A Implementation Details

This appendix specifies the candidate preparation, batch assembly, and joint optimization procedures used by Algorithm 1, and the sampling and labeling protocol for the subgoal semantic audits reported in the main text.

Training configuration. Natural solving, subgoal extraction, and subgoal-guided solving all run in thinking mode at a sampling temperature of 1.0, with at most 16 candidates evaluated per source problem. Natural and subgoal-guided solver rollouts share the per-stage maximum response length (16,384 tokens at steps 0–219, 32,768 at steps 220–349, 49,152 at steps 350–449, and 65,536 at steps 450–499), while subgoal extraction uses a completion cap of 16,384 tokens during the 16K stage and 32,768 tokens from step 220 onward. The shared policy is jointly updated using Eq. (8), with a default hinter reward scale of $\alpha = 0.3$. Section 4.3 compares this setting with $\alpha = 0.1$ and $\alpha = 0$ at 350 training steps. Through step 349, the learning rate follows the cosine schedule in Section 4.1, defined over a 500-step horizon. The 48K continuation (steps 350–449) holds the learning rate at 5×10^{-7}, and the 64K continuation (steps 450–499) holds it at 1×10^{-6}.

A.1 Candidate Validation and Guided-Rollout Budget

Prefix extraction and format validation. For each all-incorrect problem, the hinter receives the original question and the first 16,384 thinking tokens of each failed response. The primary output and structured JSON retry follow the verbatim templates in Appendix B.1. An extraction is placed in $\mathcal{U}_{\text{inv}}(q)$ if it is truncated, fails parser validation, contains an empty subproblem, or exceeds the 2,048-character subproblem bound. Explicit `NONE` abstentions are valid no-ops: they trigger neither guided rollouts nor hinter gradient updates.

Semantic representatives and reserve candidates. We first collapse exact string duplicates after whitespace normalization. The live policy then groups the remaining candidates using $\mathcal{P}_{\text{dedup}}$ (queried at temperature 1.0 in thinking mode under a 16,384-token cap, with a parallel non-thinking structured-JSON fallback capped at 2,048 tokens) into clusters $\{\mathcal{C}_1, \ldots, \mathcal{C}_M\}$, with a concise representative $z_m^{(0)}$ for each cluster. Equivalence requires the same mathematical question or task under the same assumptions; sharing a numerical answer is insufficient. Other members $z_m^{(1)}, z_m^{(2)}, \ldots$ remain available as reserve candidates. They retain their original text, extraction trajectories, and cluster membership.

Backfill within the candidate budget. Each source problem may submit at most $Z_{\max} = G$ candidates for guided solving. We first include one representative from each cluster. If this leaves unused capacity and reserve candidates remain, we append candidates in passes over the clusters: pass $r \geq 1$ adds at most one available $z_m^{(r)}$ from each cluster before any cluster contributes a candidate from pass $r+1$. Empty reserves are skipped, and the procedure stops when $Z_{\max}$ candidates have been admitted or all reserves are exhausted. The resulting list $\{\hat{z}_{q,k}\}_{k=1}^{K_q}$ satisfies $K_q \leq Z_{\max}$. Because at most one subgoal is extracted per failed response and $Z_{\max} = G$, the number of clusters cannot exceed the candidate cap in this configuration.

This backfill reuses already generated alternative formulations when the budget permits; it requires no additional hint generation. For example, when the budget permits, two clusters containing $(z_1^{(0)}, z_1^{(1)}, z_1^{(2)})$ and $(z_2^{(0)}, z_2^{(1)})$ yield the ordered list $(z_1^{(0)}, z_2^{(0)}, z_1^{(1)}, z_2^{(1)}, z_1^{(2)})$. Candidates within one cluster may elicit

different solver outcomes, so each submitted candidate receives its own group of G guided responses. Each submitted candidate retains its cluster label throughout verification and batch selection.

Hinter feedback eligibility. Each evaluated candidate remains linked to its extraction trajectory $u_{q,k}$, which receives the reward in Eq. (6) regardless of whether its guided group is selected for solver training. Unevaluated candidates and valid abstentions receive no hinter update. Format-invalid trajectories receive $-\alpha$, with at most $|\mathcal{U}_{\text{scored}}|$ such trajectories admitted to the hinter pool, sampled uniformly without replacement when necessary.

A.2 Ordering Guided Groups for the Solver Batch

Only completed guided groups with $0 < p_{\text{guide}} < 1$ enter the guided selection pool. Their source problem and cluster label identify a semantic cluster $(q, \mathcal{C}_m)$. Selection fills at most $B_{\text{rem}} = B - |\mathcal{G}^{\text{mix}}_{\text{nat}}|$ solver slots after retaining natural mixed groups:

1. **Order candidates within each cluster.** Sort its mixed groups by ascending p_{guide}. Assign indices $d = 0, 1, \ldots$ in that order, so $d = 0$ denotes its lowest-pass-rate mixed group.
2. **Visit source problems within a coverage round.** For a fixed index d, order each source problem's available clusters using a deterministic hash with a fixed seed, independently of pass rate. Visit source problems in passes, ordered by the ascending pass rate of their head candidate, admitting at most one index-d group per source problem in each pass.
3. **Advance only after exhausting the current round.** Complete index d across all source problems before proceeding to $d + 1$, stopping immediately if the solver budget is filled. Thus every available first group has priority over every second group, even when a second group has a lower pass rate.

Each selected group contributes all G solver trajectories. If guided mixed groups cannot fill the remaining budget, unused natural groups fill the residual slots, maintaining $N_{\text{solve}} = B \times G$.

A.3 Joint Objective and Loss Normalization

Hinter advantage scale. The baseline in Eq. (7) is computed over the global hinter pool gathered across all data-parallel ranks. If $q_+ \in [0, 1]$ denotes the fraction of positive $(+\alpha)$ extractions in $\mathcal{B}_{\text{hint}}$, the batch mean is $\bar{r}^{\text{hint}} = \alpha(2q_+ - 1)$, yielding exact centered advantages

$$\hat{A}^{\text{hint}}_{+} = +2\alpha(1 - q_+), \qquad \hat{A}^{\text{hint}}_{-} = -2\alpha q_+. \tag{9}$$

Thus α scales both positive and negative hinter advantages without being canceled by standard-deviation normalization.

Expanded role objectives. The solver pool contains B groups of G responses $o_{g,i}$, with each group conditioned on a natural or subgoal-guided prompt q_g. The hinter pool contains extraction trajectories u_m

generated from prompts x_m^{hint}. Solver advantages follow Eq. (1), with rewards obtained by verifying each response against the original problem's answer; hinter advantages follow Eq. (7). The role-specific token-level clipped surrogate objectives are

$$\begin{aligned}\mathcal{J}_{\text{solve}}(\theta) &= \frac{1}{T_{\text{solve}}} \sum_{g=1}^{B} \sum_{i=1}^{G} \sum_{t=1}^{|o_{g,i}|} \min\Big\{\rho_{g,i,t}(\theta)\hat{A}_{g,i,t}^{\text{solve}},\ \text{clip}\Big(\rho_{g,i,t}(\theta), 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}}\Big)\hat{A}_{g,i,t}^{\text{solve}}\Big\}, \\ \mathcal{J}_{\text{hint}}(\theta) &= \frac{1}{T_{\text{hint}}} \sum_{m=1}^{N_{\text{hint}}} \sum_{t=1}^{|u_m|} \min\Big\{\rho_{m,t}^{\text{hint}}(\theta)\hat{A}_{m,t}^{\text{hint}},\ \text{clip}\Big(\rho_{m,t}^{\text{hint}}(\theta), 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}}\Big)\hat{A}_{m,t}^{\text{hint}}\Big\},\end{aligned} \tag{10}$$

with token-level importance ratios $\rho_{g,i,t}(\theta) = \frac{\pi_\theta(o_{g,i,t}|q_g, o_{g,i,<t})}{\pi_{\theta_{\text{old}}}(o_{g,i,t}|q_g, o_{g,i,<t})}$ and $\rho_{m,t}^{\text{hint}}(\theta) = \frac{\pi_\theta(u_{m,t}|x_m^{\text{hint}}, u_{m,<t})}{\pi_{\theta_{\text{old}}}(u_{m,t}|x_m^{\text{hint}}, u_{m,<t})}$, and total active response tokens $T_{\text{solve}} = \sum_{g=1}^{B}\sum_{i=1}^{G}|o_{g,i}|$ and $T_{\text{hint}} = \sum_{m=1}^{N_{\text{hint}}}|u_m|$. In implementation, Eq. (8) is optimized in a single unified forward–backward pass under a shared global per-token loss reducer over $T_{\text{total}} = T_{\text{solve}} + T_{\text{hint}}$ tokens by multiplying $\hat{A}_{g,i,t}^{\text{solve}}$ and $\hat{A}_{m,t}^{\text{hint}}$ by the exact pool-balancing scalars

$$s_{\text{solve}} = \frac{T_{\text{total}}}{N_{\text{solve}} + N_{\text{hint}}} \cdot \frac{N_{\text{solve}}}{T_{\text{solve}}}, \qquad s_{\text{hint}} = \frac{T_{\text{total}}}{N_{\text{solve}} + N_{\text{hint}}} \cdot \frac{N_{\text{hint}}}{T_{\text{hint}}}. \tag{11}$$

A.4 Subgoal Semantic Audit and Blind Labeling Protocol

We audit solver-visible candidate subgoals every 40 training steps from step 19 to step 379 for `FORGE`, answer-anchored joint RL (*w/ a**), and solver-only updates (*w/o Hinter RL*; audited through step 339). The pass-rate-stratified audit in Figure 6c covers mixed groups from steps 220–270.

1. Exact hint recovery and stratified inverse-probability sampling. In every audited cell, we recover the exact solver-visible subgoal text appended in the `(Hint: Try to solve the following subproblem first; ...)` scaffold from each $G = 16$ guided rollout group. We construct two complementary sampling views at every (method, step) cell under a fixed random seed:

- **Stratified candidate pool view:** To prevent any single pass-rate regime from dominating the candidate pool estimate, we partition all evaluated candidate subgoal groups in a given (method, step) cell into five mutually exclusive strata according to their empirical guided pass rate p_{guide} over $G = 16$ rollouts: (i) *Zero* (0/16 correct, quota up to 16 groups); (ii) *Low* (1/16–4/16 correct, quota up to 12 groups); (iii) *High* (5/16–15/16 correct, quota up to 8 groups); (iv) *Perfect* (16/16 correct, quota up to 8 groups); and (v) *Invalid / Unscored* (quota up to 8 groups). Within each stratum s of population size N_s and sampled size $n_s = \min(N_s, \text{quota}_s)$, groups are drawn uniformly without replacement and assigned exact Horvitz–Thompson inverse inclusion weights $w_s = N_s/n_s$ to reconstruct unbiased population proportions across the full candidate pool.
- **Admitted training batch view:** Independently of the candidate-pool sample, we draw a uniform simple random sample of up to 12 subgoal groups from the subgoal-guided groups actually admitted into the 128-group solver training batch $\mathcal{B}_{\text{solve}}$ at that step.

Deduplicating overlapping records between the two views yields **1,215** unique candidate records (**1,365** view memberships) across the 40-step grid from step 19 to step 379. From step 220 onward, the answer-anchored baseline applies online leakage filtering prior to guided solving, so its audited candidate pool reflects the surviving solver-visible hints. In addition, to audit the relationship between downstream guided pass rate $p_{\text{guide}} = k/16$ and intrinsic subgoal quality during the 32K stage (Figure 6c), we apply deterministic

water-filling stratified sampling across all 15 mixed pass-count strata $k \in \{1, \ldots, 15\}$ over steps 220–270 (up to 24 mixed subgoals per step without replacement), yielding **1,205** unique mixed subgoals across 409 source problems (**1,186** scored after excluding 19 `uncertain` items) evaluated under the identical blind rubric and calibration protocol below.

2. Mutually exclusive seven-category taxonomy. Each subgoal record is classified into one of seven mutually exclusive semantic categories ordered by strict priority (Table 5). Crucially, our rubric separates *mathematical plausibility* from *answer leakage*: a hint that is mathematically true is still classified as a leak if it supplies the unknown target value or a trivial arithmetic encoding of $a^\star$ as given information. Conversely, a hint that poses a genuinely non-trivial intermediate subproblem whose uncomputed solution happens to coincide with $a^\star$ (without stating $a^\star$ as an input or assumption) is not penalized as leakage. We use two aggregate indicators in these audits: **Constructive Scaffolds** (*Useful Subgoal* + *Method / Setup*) and **Shortcut Leakage** (*Direct Answer Leak* + *Answer Proxy*). Figure 3a plots the constructive share; Figures 9–10 provide the aggregate leakage comparison and the full category distributions.

Table 5. Mutually exclusive seven-category semantic taxonomy used for source-blinded subgoal quality auditing (ordered by priority).

Priority & Key	Category Label	Operational Definition & Boundary Criteria
1. `direct_answer`	Direct Answer Leak	Explicitly states the final ground-truth value $a^\star$, an equivalent complete answer object, or supplies the full answer object as a given input and merely asks for a trivial transformation (e.g., giving side lengths $(9, 8, 7)$ when $a^\star = (9, 8, 7)$ and asking for the perimeter, or appending `\boxed{`$a^\star$`}`).
2. `answer_proxy`	Answer Proxy	Does not write $a^\star$ verbatim, but replaces the substantive mathematical work of the original problem with a trivial arithmetic expression or encoding that directly evaluates to $a^\star$ (e.g., replacing an extremal graph proof on 2022 vertices with "Calculate $\lfloor 3 \times 2022/4 \rfloor + 1$"). Requires explicit mathematical verification that the proxy expression evaluates to $a^\star$.
3. `useful_subgoal`	Useful Subgoal	Formulates a legitimate, relevant smaller mathematical subproblem (e.g., an intermediate lemma, auxiliary count, geometric configuration, or uncomputed integral) where solving the subproblem or bridging from it to q still requires substantive mathematical reasoning.
4. `method_setup`	Method / Setup	Proposes a constructive modeling direction, coordinate setup, auxiliary variable, parameterization, or proof strategy without crystallizing a self-contained smaller subproblem, while leaving non-trivial derivation to the solver.
5. `restatement`	Restatement	Merely repeats or algebraically rephrases the original question q without introducing decomposition, intermediate stepping stones, or new structural information.
6. `irrelevant_wrong`	Irrelevant / Wrong	Contains mathematical falsehoods, incompatible extra assumptions, unsupported constructions, or tasks unrelated to the original problem.
7. `uncertain`	Uncertain	Source problem has missing diagrams, ambiguous phrasing, or suspect reference labels that prevent reliable classification.

3. Source-blinded LLM judging, calibration gate, and mathematical verification. To conceal method and training stage from the judge, records are shuffled into mixed mini-batches of four under the same rubric and model settings. The judge model (`Qwen3.6-27B` in thinking mode, 16,384-token generation cap, temperature 1.0, top-$p = 0.95$) receives only the anonymized record payload `{"id", "question", "gold", "hint"}`; all identifying fields (`method`, `step`, and the empirical guided pass rate) are stripped prior to inference. We validate the judge on an eight-item calibration set covering the taxonomy prior to full labeling, followed by

secondary mathematical verification on all flagged shortcut candidates and a stratified random audit subset to ensure boundary consistency (e.g., verifying whether candidate arithmetic expressions actually evaluate to $a^\star$ and ensuring consistent treatment of equivalent subproblems across steps). Uncertainty intervals for all category shares are estimated via 4,000-iteration source-problem cluster bootstrap combined with an effective-source-count Wilson score envelope. The verbatim audit prompt is given in Appendix B.5.

A.5 Science and STEM Training Data and Evaluation

Training data curation. We start from answer-verified Nemotron-RL-knowledge-mcqa questions with ten valid options and eight completed rollouts from a reference policy. After excluding known non-STEM categories, we use `Qwen3.6-27B` to annotate 52,536 candidates with at most five correct rollouts and 12,552 with six or seven, recording discipline, reasoning type, difficulty, and question quality. We retain clean, nonclinical STEM questions that require multistep quantitative or deep conceptual reasoning, have a difficulty rating of at least 2 on a five-point scale, and have a mean reference rollout length between 2,000 and 50,000 tokens. We screen normalized question stems for exact and near duplicates against GPQA-Diamond, SuperGPQA, HLE, and MMLU-Pro and within the training pool. Of the 23,342 retained questions, 16,689 have at most five correct reference rollouts, 3,653 have six, and 3,000 have seven.

We stratify prompt assignment by discipline and number of correct reference rollouts. Half of the questions (11,671) request the option letter on the final line using either `Answer: A/B/C/D/E/F/G/H/I/J` or `Answer: X`, and the other half request the letter inside `\boxed{}`. The training reward checks the extracted letter against the gold option. Each question includes a reference solution for SAGE and HATCH from `Qwen3.6-27B` whose extracted final answer matches the gold option.

Evaluation datasets. GPQA-Diamond contains 198 four-option questions, with option order shuffled using seed 0. SuperGPQA-STEM-Hard contains 5,208 hard ten-option questions from science and engineering fields, excluding mathematics. HLE-Science-MC contains 301 text-only multiple-choice questions in biology and medicine, chemistry, computer science and AI, engineering, and physics. MMLU-Pro-STEM contains 4,527 test questions from physics, chemistry, biology, computer science, and engineering.

Evaluation protocol. We evaluate with EvalScope [33] using zero-shot prompts without hints, with thinking enabled and a generation limit of 32,768 tokens. We sample at temperature 1.0, top-$p = 0.95$, and top-$k = 20$, generating five responses per question for GPQA-Diamond, four for HLE-Science-MC, and one each for SuperGPQA-STEM-Hard and MMLU-Pro-STEM. GPQA-Diamond and MMLU-Pro-STEM use the Artificial Analysis multiple-choice prompt, which requests a final answer such as `Answer: A`. SuperGPQA-STEM-Hard uses its chain-of-thought prompt, which requests `ANSWER: [LETTER]` on the final line. HLE-Science-MC uses the HLE multiple-choice system prompt, which requests `Explanation:`, `Answer:`, and `Confidence:` fields. GPQA-Diamond and MMLU-Pro-STEM use the Artificial Analysis regular-expression extractor with fallbacks for boxed and free-form answers. SuperGPQA-STEM-Hard and HLE-Science-MC can recover the option letter from the choice text. Each benchmark scores the extracted letter by exact match against the gold option. We average correctness over the sampled responses for each question and report the mean across questions as a percentage.

B Prompt Templates

This appendix records the verbatim prompt templates used by FORGE during online RL training, by the generative equivalence judge for **ArXivMath_0526** and **ArXivMath_0626**, and by the source-blinded subgoal semantic audit. Training uses the reference-free, question-only subgoal extraction prompt $\mathcal{P}_{\text{hint}}(q, \tau_i)$ (with its primary line format and JSON fallback format), the policy-driven semantic deduplication prompt $\mathcal{P}_{\text{dedup}}(q, \{z_i\})$, and the subgoal-guided solving prompt $\mathcal{P}_{\text{solve}}(q, \hat{z}_{q,k})$.

B.1 Reference-Free Subgoal Extraction

Prompt Template for Reference-Free Subgoal Extraction ($\mathcal{P}_{\text{hint}}$)

You are extracting a subgoal QUESTION to append as a hint. Do not answer the question or solve the original problem.

=== ORIGINAL PROBLEM ===
`{question}`

=== FAILED ROLLOUT `{rollout_index}` **THINKING (wrong; shows what the policy attempts) ===**
`{truncated_failed_thinking_trace}`

You are selecting a subgoal QUESTION to append as a hint to the original problem. The solver sees the original problem plus your question, never your reasoning or an expected answer. Select a question from what the failed trajectory attempted, computed, or might usefully reconsider. The trajectory may be wrong. You do not need to verify its intermediate answers or solve either the subgoal or the original problem before proposing a question.

The sole semantic exclusion is supplying the original problem's final answer inside the solver-visible question. Apply this check to the QUESTION itself:

- Do not state the original unknown target's correct value as a given condition, assumption, test candidate, or completed requested construction.
- Do not insert the complete gold answer, including an equivalent expression or full constructed object, and merely ask for its value, simplification, digits, verification, or another transformation. Renaming it does not remove the leak.
- Do not refer to a gold/standard answer or private material to supply the answer.
- A literal matching gold may be an ordinary problem constant, count, index, or coefficient. Its presence alone is NOT leakage. Reject only when its role is to provide the original answer or an answer object to read or transform.
- The answer that the solver would COMPUTE may equal gold. That is allowed; asking for a result is different from giving that result as an input.

Special cases, auxiliary assumptions defining a case, small examples, yes/no questions, proof questions, strategy questions, easy arithmetic, restatements, and partial computations are all allowed if they do not provide the final answer. Do not require a unique short numerical answer. Do not reject because you cannot prove relevance, cannot verify an answer, or think the question is too easy. Whether it helps this policy is measured later by rollouts that append this subgoal as a hint.

For example, if the original asks how many configurations work and gold is 2, asking about the possibilities for two objects is allowed. Stating that exactly two configurations work and asking to count them gives the answer and is forbidden. If the original asks for a total, computing one component is allowed; supplying the final total and asking for the remaining component is forbidden.

Write the question so it can be read alongside the original problem; include any auxiliary setup it needs. Do not append the answer or a solution. If a proposed question leaks the original answer, choose another question. Output `NONE` only if you cannot formulate a question without providing the answer. Check briefly and emit the question; do not spend the budget solving it.

[Primary Line Format]
Output ONLY `NONE` or:
`SUBPROBLEM: <the question, without its answer>`

[JSON Fallback Format (triggered only upon primary format failure)]
Output ONLY one JSON object:
`{"subproblem": "<the question, without its answer>"}`
If no question can be proposed without supplying the answer, output `{"subproblem": null}`.

B.2 Policy-Driven Semantic Deduplication

Prompt Template for Policy-Driven Semantic Deduplication ($\mathcal{P}_{\text{dedup}}$)

You are deduplicating extracted subgoals before expensive rollout verification.
Use the original problem to understand each question or task (including proofs and strategies).
Do not solve the original problem and do not rewrite any candidate.

=== ORIGINAL PROBLEM ===
`{question}`

=== CANDIDATES ===
```
C0:
SUBPROBLEM: {subproblem_0}
C1:
SUBPROBLEM: {subproblem_1}
...
```

=== DEDUPLICATION RUBRIC ===

- Group candidates only when they ask the same question or task under the same conditions.
- Merge paraphrases, notation-only rewrites, and versions differing only by irrelevant background.
- The same numerical answer alone is NOT evidence of duplication.
- Keep candidates separate when they target different variables, different intermediate stages, different conditions, or different levels of generality.
- When uncertain, keep them separate.
- Choose as `KEEP` a concise, clear representative understood alongside the original problem. Do not judge usefulness or solve the task.

Return duplicate clusters in the required JSON schema.
For example, `C2` representing duplicates `C5` and `C8` is:
`{"clusters":[{"keep":2,"duplicates":[5,8]}]}`
If there are no duplicate clusters, return `{"clusters":[]}`.

B.3 Subgoal-Guided Solving

Prompt Template for Subgoal-Guided Solving ($\mathcal{P}_{\text{solve}}$)

```
{question}

(Hint: Try to solve the following subproblem first; it may help solve the problem above:
{subproblem})
```

B.4 Generative Judge Prompt for ArXivMath_0526 and ArXivMath_0626

For **ArXivMath_0526** and **ArXivMath_0626**, responses with a closed `\boxed{}` prediction that are not directly matched by the rule-based symbolic verifier (`math_equal`) are evaluated by `Qwen3.6-27B-RM` using the equivalence-checking prompt below (translated from the original Chinese template used in our evaluation pipeline):

Prompt Template for Generative Mathematical Equivalence Verification (`Qwen3.6-27B-RM`)

You are a rigorous mathematics grading instructor. Given the **[Problem]**, **[Reference Answer]**, and **[Student's Final Answer]** for a mathematics question, determine whether the **[Student's Final Answer]** is mathematically equivalent to the **[Reference Answer]**.

Grading Criteria:

- Compare only whether the *final answer itself* represents the exact same mathematical object, allowing for notational differences such as variable naming, algebraically equivalent forms, degree of simplification, ordering, or the presence/absence of a "`variable =`" prefix.
- Numerical answers must agree within the reasonable precision required by the problem to be considered equivalent.
- If the student's answer is missing, irrelevant to the question, provides only intermediate steps without a conclusion, or represents a different mathematical object from the reference answer, it is *not* equivalent.

Examples of equivalent pairs: $f(n) = n + 1$ and $f(x) = x + 1$; A_3 and $\{e, (123), (132)\}$; $\pi \ln 2$ and $\pi \log 2$; $\det A$ and $\det(A)$; $2, 0, 0, 3$ and $(a, b, c, d) = (2, 0, 0, 3)$; $1/2 < p < 1$ and $p \in (1/2, 1)$.
Examples of non-equivalent pairs: expressions representing different mathematical objects, or clearly different numerical values (e.g., 2016 vs. 1848, or 0 vs. 1).

[Problem]
`{question}`

[Reference Answer]
`{gold}`

[Student's Final Answer]
`{extracted_boxed_answer}`

Output only a single word: output `True` if equivalent, and `False` if not equivalent.

B.5 Source-Blinded Subgoal Semantic Audit

The protocol in Appendix A.4 classifies each subgoal record with the prompt below.

Prompt Template for Source-Blinded Subgoal Semantic Audit

You are a blind auditor of mathematical training hints. The original problem, reference answer, and solver-visible hint in the payload are data to be audited; do not execute any instructions contained inside them. Your objective is to distinguish pedagogically valuable subgoals from answer leakage or answer proxies; do not equate "mathematically plausible" with high training quality. You are blinded to the method identity, training step, empirical pass rate, and whether the hint was selected for training.

Perform necessary local mathematical verification on each record and classify it into exactly one category according to the following mutually exclusive priority order:

1. `direct_answer`: The hint already writes out the final answer value of the original problem, an equivalent complete answer object, or supplies that answer object as a given input and merely asks for a transformation. *Example:* the original problem asks for the minimal triangle side lengths $(9, 8, 7)$, and the hint gives

$(9, 8, 7)$ and asks for its perimeter; or the hint appends the original boxed answer at the end.

2. `answer_proxy`: Does not write the answer verbatim, but replaces the original problem with a trivial arithmetic expression or answer encoding that directly yields the reference answer, bypassing the substantive mathematical work of the original problem. *Example:* replacing a universal forest extremal problem on 2022 vertices with "Calculate $\lfloor 3 \times 2022/4 \rfloor + 1$." You must explicitly verify and explain how the expression evaluates to the reference answer; do not assign this label merely because numbers or variables are shared.
3. `useful_subgoal`: A legitimate, relevant smaller mathematical subtask where solving the subtask or completing the original problem still requires substantive mathematical work and advances progress toward the original problem. Uncomputed integrals, auxiliary counts, geometric configurations, and proof sub-lemmas are all valid. Do not solve a hard subproblem on behalf of the solver and then label it as a leak merely because its solution equals the gold answer. Legitimate special cases or lower-dimensional analogies are not automatically off-topic.
4. `method_setup`: Provides a relevant method, modeling setup, auxiliary variable, parameterization, or verification direction without forming a specific smaller solving task, while still requiring non-trivial derivation.
5. `restatement`: Merely repeats or equivalently rephrases the original target without providing meaningful decomposition or new information. Do not assign this label solely because the target looks similar; judge whether it genuinely reduces the reasoning workload.
6. `irrelevant_wrong`: Mathematically incorrect, substantively unrelated to the original problem, imposes false additional conditions, or proposes an unsupported construction. Difficulty or inability to immediately verify does not imply incorrectness.
7. `uncertain`: Insufficient material—such as missing diagrams, problem ambiguity, or a suspect reference answer—prevents reliable determination of the primary category. Flag source issues separately; if `direct_answer` is structurally clear, you may still classify it as disclosure rather than `uncertain`.

Boundary Rules: When a hint asks to "prove a quantitative assertion containing the unknown target of the original problem," check whether the unknown target value is already supplied as a given constant inside the assertion. If reading or computing given coefficients in the assertion directly reveals the final answer, record `answer_proxy` and explain; if the hint asks the solver to solve a still-difficult subtask where the answer is not supplied as given, do not classify as leakage. Separate direct answer objects from proof completeness: supplying the full constructed answer object and asking for its perimeter remains `direct_answer`. Do not blindly trust the reference answer when checking consistency; mark discrepancies via `source_quality`.

Output ONLY a JSON object with structure `{"results": [...]}` without markdown wrappers, with one entry per `id`:

```
{
  "id": "<input_id>",
  "category": "direct_answer | answer_proxy | useful_subgoal | method_setup | restatement |
irrelevant_wrong | uncertain",
  "source_quality": "sufficient | missing_figure | ambiguous | suspect_reference | uncertain",
  "remaining_work": "trivial | nontrivial | substantial | uncertain",
  "confidence": "high | medium | low",
  "provided_information": "<what known info the hint supplies, <=80 chars>",
  "reason": "<specific mathematical justification, <=120 chars>"
}
```

C Detailed Empirical Audits and Diagnostics

To ensure full empirical transparency and reproducibility, this section provides the complete underlying numerical data, category breakdowns, correlation statistics, and 95% cluster-bootstrap confidence intervals for the diagnostic figures in Section 1, Section 3, and Section 4:

- **Diagnostic audit of $0/16$ failed natural rollout prefixes and reference-solution hinting versus `FORGE` across steps 19–219 (Table 6 and Table 7):** Source-blinded failure-stage classification of 250 failed natural thinking prefixes from $0/16$ (all-incorrect) groups together with downstream subgoal quality and problem-unlocking rates (underlying Figure 1b), paired with surface outcomes and deep semantic audit shares of admitted hinted groups in SAGE versus `FORGE`.
- **Longitudinal subgoal semantic audit on the 40-step grid through step 379 (Table 8):** Complete Horvitz–Thompson inverse-probability-weighted candidate pool proportions across all seven semantic categories, aggregate *Constructive Scaffolds* (95% CI), aggregate *Shortcut Leakage* (95% CI), and admitted training batch constructive shares for `FORGE`, *w/ Ground-Truth* $a^\star$, and *w/o Hinter RL* ($\alpha = 0$), underlying Figure 3a and the supplementary analyses in Appendix C.1.
- **Pass-rate-stratified subgoal quality and correlation suite across steps 220–270 (Table 9 and Table 10):** Exact per-stratum ($k/16 \in \{1/16, \ldots, 15/16\}$ and coarse regimes) sample counts, category counts, mean quality scores, and constructive shares across all $1,205$ audited mixed subgoals ($1,186$ scored across 409 source problems), together with global Spearman ρ, Pearson r, within-(q, step) demeaned correlations, and paired $> 1/16$ minus $1/16$ differences across all seven sensitivity specifications, corresponding to Figure 6c.
- **Hinter reward scale ablation and likelihood-ratio transfer-weight diagnostics (Table 11, Table 12, and Appendix C.2):** Complete seven-benchmark Avg@32 and Pass@32 scores for $\alpha \in \{0, 0.1, 0.3\}$ (underlying Figure 6a–b), together with length-bin means (ρ_i vs. unnormalized $\sum_t \Delta \log \pi_t$), prefix quartile shares, and length-dilution diagnostics for likelihood-ratio transfer weighting (Figure 11a–b).
- **Semantic curriculum selection A/B rollout and periodic 32K benchmark evaluations (Table 13):** Complete online rollout window summaries, admitted batch semantic-cluster coverage and repeat statistics, and per-benchmark periodic 32K Pass@1 and Pass@k trajectories with their problem-weighted micro averages across steps 229–269, corresponding to Figure 4a–c.
- **Controlled 350-step problem mastery and test-time scaling (Table 14):** Six-stratum mastery counts and paired head-to-head unlocking across all 312 benchmark problems, together with the unbiased micro-average Pass@k curves ($k \in \{1, 2, 4, 8, 16, 32\}$) and micro-average accuracy within generation budgets 8K–80K.
- **Extended test-time Pass@k scaling up to $k = 512$ (Table 15):** Per-benchmark and micro-average Pass@k percentages and solved problem counts for `FORGE` and `Qwen3.5-4B-Base` across $k \in \{1, 2, 4, 8, 16, 32, 64, 128, 256, 512\}$, corresponding to Figure 8.
- **Training time of the same 350-step runs (Figure 12):** Cumulative GPU hours of the rollout phase and the parameter update.

Table 6. Source-blinded diagnostic audit of 250 failed natural thinking prefixes $\tau_i = \text{think}(o_i)$ (mean length 9,394 tokens; 186 completed with wrong answers, 64 truncated at 16,384 tokens) from 0/16 (all-incorrect) groups across steps 19, 59, 99, 139, 179, 219, underlying Figure 1b. The prefix judge (`Qwen3.8-27B`) is blinded to the extracted subgoal z_i and guided pass rate p_{guide}. Part A reports aggregate and per-step Horvitz–Thompson candidate-pool shares; Part B cross-tabulates each failure mode with downstream subgoal quality, same-step mixed-group unlocking ($0 < p_{\text{guide}} < 1$), and unassisted solving mastery by Epoch 3.

Part A: Aggregate & Per-Step Distribution of 0/16 Failed Natural Prefixes (Figure 1b)							
		Structural Readiness (%)		**Mutually Exclusive Failure Mode (Sum = 100%)**			
Sampling View / Checkpoint	**Audited N**	**Valid Setup**	**Extractable Bottleneck**	**Midway Impasse**	**Late Execution Slip**	**Early Misdirection**	**Premature Guessing**
Overall Candidate Pool (HT-Weighted)	**199**	**91.9**	**89.1**	**71.6**	**12.6**	11.4	4.4
Admitted Training Batch View	72	91.5	89.5	64.0	20.8	9.9	5.3
All Audited Prefixes (Unweighted)	250	92.0 (230)	88.4 (221)	67.2 (168)	18.0 (45)	10.0 (25)	4.8 (12)
Step 19 (HT Pool / All)	38/46	87.2	84.6	66.2	16.4	17.4	0.0
Step 59 (HT Pool / All)	30/40	100.0	97.8	82.4	10.5	0.4	6.6
Step 99 (HT Pool / All)	32/38	89.0	94.2	80.4	3.3	10.5	5.8
Step 139 (HT Pool / All)	34/45	91.1	88.3	64.1	14.3	17.4	4.2
Step 179 (HT Pool / All)	29/35	100.0	92.8	80.3	18.2	1.5	0.0
Step 219 (HT Pool / All)	36/46	87.0	81.3	67.1	9.3	13.0	10.6
Part B: Downstream Subgoal Conversion & Cross-Epoch Unassisted Mastery by Failure Mode (Figure 1b)							
Prefix Failure Mode	**Scored N**	**Valid Setup (%)**	**Extractable Bottleneck (%)**	**Constructive Subgoal (%)**	**Concrete Subgoal (%)**	**Unlocked Mixed (1–15/16, %)**	**Solved Unassisted by Epoch 3 (%)**
Midway Reasoning Impasse	167	100.0	97.6	75.5	52.7	55.1	73.2
Late Execution Slip	43	100.0	90.7	**93.0**	**67.4**	**83.7**	**80.0**
Early Misdirection	25	36.0	44.0	76.0	56.0	60.0	68.0
Premature Guessing	12	66.7	58.3	75.0	25.0	66.7	66.7

Table 7. Audit of admitted hinted groups in reference-solution hinting (SAGE, $n = 134$ across steps 19, 59, 99, 139, 179, 219, broken down by progressive hint level $L \in \{1, 2, 3\}$) versus `FORGE` ($n = 72$ admitted guided groups across the same six checkpoints). Surface columns report the rollout outcome distribution of admitted hinted groups; deep audit columns report the source-blinded semantic classification (`Qwen3.8-27B`) on mixed (1–15/16) groups; and the final columns report the end-to-end net yield out of all admitted hinted groups.

		Surface Batch Outcomes (%)				**Deep Audit of Mixed Hints (%)**				**Net Yield / All Hinted (%)**	
Method / Stratum	**Admitted N**	**Mixed (1–15)**	**Still 0/16**	**All 16/16**	**Zero-Grad Waste**	**Constructive**	**Concrete Subgoal**	**Method / Setup**	**Leak / Spoiler**	**Net Constructive**	**Net Concrete Subgoal**
SAGE Level 1 (26.1%)	35	94.3	0.0	5.7	5.7	75.8	18.2	57.6	3.0	71.4	17.1
SAGE Level 2 (25.4%)	34	91.2	0.0	8.8	8.8	71.0	9.7	61.3	22.6	64.7	8.8
SAGE Level 3 (48.5%)	65	26.2	66.2	7.7	73.8	52.9	5.9	47.1	41.2	13.8	1.5
SAGE Overall (w/ Ref. Sol.)	**134**	60.4	32.1	7.5	39.6	69.1	12.3	56.8	18.5	41.8	7.5
`FORGE` **(All Steps 19–219)**	**72**	**100.0**	**0.0**	**0.0**	**0.0**	**80.6**	**45.8**	34.7	**1.4**	**80.6**	**45.8**
`FORGE` **(Late Steps 139–219)**	**36**	**100.0**	**0.0**	**0.0**	**0.0**	77.8	**52.8**	25.0	**0.0**	77.8	**52.8**

Table 8. Longitudinal subgoal semantic audit from step 19 to step 379 (1,215 records), underlying Figure 3a and Figure 10. Pool columns report inverse-probability-weighted shares (%); the last column reports the admitted-group Constructive share (95% source-cluster bootstrap confidence intervals). The answer-anchored baseline (*w/ a**) applies online leakage filtering from step 220 onward.

		Candidate Pool Categories (%)							Pool Aggregates (95% CI)		Admitted
Method	**Step**	**Useful**	**Setup**	**Restate**	**Wrong**	**Uncert.**	**Proxy**	**Direct**	**Constructive**	**Shortcut**	**Constructive**
FORGE	19	54.6	16.3	8.9	20.2	0.0	0.0	0.0	70.9 [45.9, 91.0]	0.0 [0.0, 20.2]	91.7 [59.6, 100.0]
	59	47.8	25.6	10.1	16.4	0.0	0.0	0.0	73.5 [45.1, 90.3]	0.0 [0.0, 24.5]	83.3 [52.8, 100.0]
	99	47.1	13.3	0.0	33.8	5.2	0.5	0.0	60.4 [35.6, 80.9]	0.5 [0.0, 22.0]	75.0 [40.9, 93.8]
	139	51.3	16.2	6.8	21.4	4.2	0.0	0.0	67.5 [41.0, 87.0]	0.0 [0.0, 22.5]	83.3 [46.9, 100.0]
	179	65.3	25.6	2.2	6.4	0.4	0.0	0.0	90.9 [62.2, 98.8]	0.0 [0.0, 26.0]	75.0 [22.3, 100.0]
	219	48.0	32.0	6.6	13.3	0.0	0.0	0.0	80.0 [51.7, 93.8]	0.0 [0.0, 24.3]	91.7 [59.6, 100.0]
	259	32.2	46.3	8.0	9.2	4.3	0.0	0.0	78.5 [48.0, 93.6]	0.0 [0.0, 27.2]	75.0 [42.7, 100.0]
	299	42.1	39.8	10.8	4.0	3.4	0.0	0.0	81.8 [50.8, 95.2]	0.0 [0.0, 27.8]	83.3 [48.7, 100.0]
	339	30.9	47.8	17.5	3.8	0.0	0.0	0.0	78.7 [52.4, 92.5]	0.0 [0.0, 21.7]	83.3 [50.7, 100.0]
	379	60.1	30.3	9.6	0.0	0.0	0.0	0.0	90.4 [63.7, 100.0]	0.0 [0.0, 23.5]	91.7 [59.6, 100.0]
w/ $a^\star$	19	65.3	4.3	2.4	28.0	0.0	0.0	0.0	69.6 [41.7, 88.5]	0.0 [0.0, 24.2]	50.0 [16.7, 81.8]
	59	41.8	33.6	15.1	6.2	1.1	2.2	0.0	75.4 [50.5, 90.2]	2.2 [0.0, 23.3]	75.0 [39.3, 100.0]
	99	53.4	23.3	6.0	11.2	6.0	0.0	0.0	76.7 [48.9, 92.3]	0.0 [0.0, 23.7]	75.0 [36.5, 94.0]
	139	46.8	16.7	22.5	10.0	0.0	3.9	0.0	63.5 [38.3, 83.0]	3.9 [0.0, 27.0]	83.3 [48.7, 100.0]
	179	54.5	17.5	16.1	4.9	0.7	0.7	5.7	72.1 [45.1, 89.5]	6.3 [0.0, 31.5]	75.0 [44.6, 100.0]
	219	47.6	9.0	17.5	21.5	2.0	2.3	0.0	56.7 [31.1, 79.1]	2.3 [0.0, 26.8]	66.7 [16.7, 93.5]
	259	66.6	24.4	1.1	7.8	0.0	0.0	0.0	91.0 [61.6, 98.5]	0.0 [0.0, 26.7]	75.0 [31.9, 95.0]
	299	79.9	2.8	6.4	10.9	0.0	0.0	0.0	82.7 [50.2, 95.8]	0.0 [0.0, 29.8]	91.7 [48.5, 100.0]
	339	76.1	17.1	0.6	6.1	0.0	0.0	0.0	93.2 [64.4, 100.0]	0.0 [0.0, 26.5]	91.7 [50.0, 100.0]
	379	68.9	13.0	10.3	7.8	0.0	0.0	0.0	81.9 [45.9, 97.3]	0.0 [0.0, 34.3]	91.7 [30.0, 100.0]
w/o Hinter RL	19	48.4	5.2	8.5	29.4	0.0	4.0	4.5	53.6 [30.5, 75.3]	8.5 [0.0, 31.9]	66.7 [32.5, 89.3]
	59	75.2	13.0	2.7	2.9	0.0	5.2	1.0	88.1 [62.1, 97.1]	6.2 [0.0, 31.0]	58.3 [26.2, 88.9]
	99	39.3	10.9	9.5	39.3	0.5	0.0	0.5	50.2 [26.3, 74.1]	0.5 [0.0, 23.6]	50.0 [18.2, 81.8]
	139	32.4	8.4	29.7	22.8	0.0	4.4	2.3	40.8 [21.4, 63.5]	6.7 [1.0, 27.8]	66.7 [23.1, 93.8]
	179	58.0	4.9	5.7	21.9	6.1	2.3	1.2	62.9 [37.4, 82.8]	3.5 [0.3, 26.8]	83.3 [40.8, 100.0]
	219	37.8	4.5	17.9	35.4	4.5	0.0	0.0	42.3 [19.7, 68.6]	0.0 [0.0, 24.4]	66.7 [26.1, 91.9]
	259	44.3	18.6	7.4	24.4	5.3	0.0	0.0	62.9 [36.6, 83.2]	0.0 [0.0, 22.9]	50.0 [19.6, 80.4]
	299	61.2	9.4	14.7	11.2	0.0	0.0	3.6	70.5 [33.9, 91.8]	3.6 [0.0, 41.8]	66.7 [28.9, 90.8]
	339	45.5	17.4	21.2	14.8	0.0	0.0	1.0	62.9 [32.1, 85.9]	1.0 [0.0, 31.7]	66.7 [23.1, 93.0]

Table 9. Pass-rate-stratified blind semantic audit across steps 220–270 (1,205 sampled mixed subgoals across 409 source problems, 1,186 scored after excluding `uncertain`; underlying Figure 6c). Quality scores map `useful_subgoal` and `method_setup` to 3, `restatement` to 2, and `irrelevant_wrong`, `answer_proxy`, and `direct_answer` to 1.

						Scored Category Counts					
Pass Count ($k/16$)	p_{guide}	**Sampled n**	**Scored**	**Uncert.**	**Mean Score (1–3)**	**Constructive (%)**	**Useful**	**Setup**	**Restate**	**Wrong**	**Proxy / Direct**
1/16	0.0625	246	241	5	2.805	88.0	101	111	11	16	2/0
2/16	0.1250	236	235	1	2.791	86.0	113	89	17	14	2/0
3/16	0.1875	190	189	1	2.788	83.6	92	66	22	6	2/1
4/16	0.2500	143	142	1	2.831	88.0	67	58	10	4	2/1
5/16	0.3125	113	111	2	2.811	86.5	47	49	9	5	1/0
6/16	0.3750	61	60	1	2.850	90.0	25	29	3	3	0/0
7/16	0.4375	61	58	3	2.690	77.6	19	26	8	2	3/0
8/16	0.5000	34	33	1	2.758	81.8	10	17	4	2	0/0
9/16	0.5625	40	39	1	2.590	69.2	16	11	8	0	3/1
10/16	0.6250	17	17	0	2.529	64.7	4	7	4	1	1/0
11/16	0.6875	15	15	0	2.600	73.3	7	4	2	1	1/0
12/16	0.7500	17	14	3	2.500	64.3	3	6	3	2	0/0
13/16	0.8125	12	12	0	2.083	41.7	3	2	3	1	1/2
14/16	0.8750	9	9	0	2.667	77.8	3	4	1	0	1/0
15/16	0.9375	11	11	0	2.636	72.7	1	7	2	1	0/0
Coarse Pass-Rate Regimes											
1/16 (Frontier)	0.0625	246	241	5	**2.805**	**88.0**	101	111	11	16	2/0
2–6/16 (Moderate)	0.125–0.375	743	737	6	**2.806**	**86.2**	344	291	61	32	7/2
7–15/16 (High)	0.438–0.938	216	208	8	2.611	72.1	66	84	35	10	10/3
Total / Overall	1/16–15/16	**1,205**	**1,186**	**19**	**2.772**	**84.1**	**511**	**486**	**107**	**58**	**19/5**

Table 10. Correlation and robustness analysis between downstream guided pass rate p_{guide} and intrinsic subgoal quality score across steps 220–270 (2,000-iteration source-question cluster bootstrap 95% confidence intervals). Across all sampling and weighting specifications, p_{guide} exhibits no positive correlation with subgoal quality.

			Global Association (p_{guide} vs. Quality Score)			Within-Problem-and-Step Controlled Comparison		
Analysis Specification	**Scored N**	**Sources**	**Spearman ρ**	**95% Cluster CI**	**Pearson r**	**Demeaned r (N_{within})**	**95% Cluster CI**	**Δ(>1/16 − 1/16) (95% CI)**
1. Primary balanced stratified sample	1,186	408	−0.101	[−0.168, −0.031]	−0.122	−0.071 (978)	[−0.154, +0.003]	−0.096 [−0.232, +0.035]
2. Uniform $k/16$-stratum reweighted	1,186	408	−0.203	[−0.322, −0.075]	−0.193	−0.071 (978)	[−0.154, +0.003]	−0.096 [−0.232, +0.035]
3. Inverse-probability population reweighted	1,186	408	−0.083	[−0.169, −0.001]	−0.095	−0.071 (978)	[−0.154, +0.003]	−0.096 [−0.232, +0.035]
4. Sufficient source material only	1,109	397	−0.095	[−0.165, −0.025]	−0.118	−0.054 (904)	[−0.134, +0.024]	−0.073 [−0.202, +0.052]
5. `uncertain` included as score 2	1,205	409	−0.108	[−0.175, −0.037]	−0.128	−0.074 (998)	[−0.152, +0.007]	−0.080 [−0.212, +0.048]
6. One candidate per $(q, \text{step}, \mathcal{C}_m)$ cluster	1,015	408	−0.102	[−0.178, −0.023]	−0.124	−0.059 (785)	[−0.145, +0.021]	−0.128 [−0.268, +0.012]
7. Admitted training batch subset	438	288	−0.066	[−0.159, +0.033]	−0.091	−0.168 (209)	[−0.344, −0.008]	−0.136 [−0.364, +0.081]

Table 11. Hinter RL ablation after 350 training steps (16K → 32K context; underlying Figure 6a–b). We report Avg@32 and Pass@32 (%) under the same evaluation protocol as Table 2. Setting $\alpha = 0$ removes direct hinter RL, while $\alpha = 0.3$ is the default for `FORGE`. Best results in each column are highlighted in **bold**. The default `FORGE` row ($\alpha = 0.3$) is shaded.

α	AIME 25		AIME 26		HMMT Feb25		HMMT Feb26		Beyond AIME		ArXivMath 0526		ArXivMath 0626		Micro Average	
	Avg@32	Pass@32	Avg@32	Pass@32	Avg@32	Pass@32	Avg@32	Pass@32	Avg@32	Pass@32	Avg@32	Pass@32	Avg@32	Pass@32	**Avg@32**	**Pass@32**
0	79.58	**96.67**	79.90	93.33	**68.02**	**93.33**	58.90	**90.91**	52.59	81.00	27.03	67.50	12.44	53.06	50.38	79.81
0.1	78.33	**96.67**	81.46	93.33	64.38	90.00	60.61	87.88	51.38	81.00	26.33	67.50	12.88	53.06	49.83	79.17
0.3	**80.31**	**96.67**	**82.71**	**96.67**	65.52	**93.33**	**62.88**	87.88	**53.38**	**84.00**	**28.75**	**70.00**	**13.90**	**63.27**	**51.60**	**82.69**

Table 12. Dual-context teacher-forced likelihood-ratio diagnostics across response length bins, prefix quartiles, and correlation views (underlying Figure 11a–b; 52 mixed subgoal groups and 179 correct trajectories at step 220, plus 438 mixed groups / 2,053 correct trajectories across steps 220–229).

Part A: Trajectory Length-Bin Divergence ($N = 179$ Correct Trajectories)

Length Bin $\lvert y_i \rvert$	Trajectories	Mean ρ_i (nats/tok)	Mean $\sum_t \Delta \log \pi_t$ (nats)	Relative ρ_i Dilution
< 2K tokens	30	0.1044	133.27	1.00× (ref.)
2K–5K tokens	61	0.0497	156.98	2.10× lower
5K–10K tokens	42	0.0222	156.83	4.70× lower
≥ 10K tokens	46	0.0132	188.75	7.88× lower
All Trajectories	**179**	**0.0280** (group mean)	**163.55** (group mean)	**7.88× bin span**

Mean transfer weight on positive hints: $\bar{w}_h = 0.914$ ($T = 0.3$), $\bar{w}_h = 0.973$ ($T = 1.0$).
Dual-context tokens scored per step: 2.53M tokens (1.24M response tokens ×2).

Part B: Correlation & Prefix Concentration Summary

Diagnostic Metric	Value	p-value / IQR
Group-level Spearman $r(\rho_h, \overline{\lvert y \rvert}_{\text{correct}})$	−0.916	$p = 1.80 \times 10^{-21}$
Group-level Spearman $r(\rho_h, \overline{\lvert y \rvert}_{\text{all}})$	−0.803	$p = 8.35 \times 10^{-13}$
Trajectory-level Spearman $r(\rho_i, \lvert y_i \rvert)$	−0.956	$p < 10^{-30}$
Within-source Spearman $r(\rho_h, \overline{\lvert y \rvert}_{\text{correct}})$	−1.000 (median)	Range [−1.00, −0.50]
Unnormalized sum Spearman $r(\sum \Delta \log \pi, \overline{\lvert y \rvert}_{\text{correct}})$	+0.340	$p = 0.0138$
First-quartile (Q_1, first 25% tokens) cumulative share (all $\lvert y \rvert$)	**91.1**% (median)	IQR [72.4%, 95.7%]
First-quartile (Q_1, first 25% tokens) share ($\lvert y_i \rvert \geq 10$K)	**86.2**% (median)	IQR [67.5%, 93.3%]

Table 13. Complete online rollout and periodic 32K benchmark evaluation comparison between selection without semantic clusters (*w/o ss*) and `FORGE` across steps 220–275 (branched from the identical step-219 checkpoint; underlying Figure 4a–c). The micro average weights each of the 263 problems equally: 30 in AIME25, AIME26, and HMMT_Feb25, 33 in HMMT_Feb26, 100 in BeyondAIME, and 40 in ArXivMath.

Part A: Online Natural Rollout & Admitted Guided Batch Summary Across Matched Step Windows

Step Window	Selector Variant	all-incorrect/ Step	Natural Mixed / Step	Cluster Coverage	Cluster Repeats	Admitted $p_{\text{guide}} \geq 2/16$
Steps 220–275 (56 steps)	*w/o ss*	15.95	90.20	92.17% (1,812/1,966)	9.94% (200/2,012)	12.41% (mean $p = 7.58\%$)
	`FORGE`	**15.18** (**−0.77**)	**91.20** (**+1.00**)	**100.0**% (**1,966/1,966**)	**2.29**% (**46/2,012**)	**34.41**% (mean $p = 11.40\%$)
Steps 240–275 (36 steps)	*w/o ss*	15.11	90.56	—	—	—
	`FORGE`	**14.31** (**−0.81**)	**91.72** (**+1.17**)	—	—	—
Steps 260–275 (16 steps)	*w/o ss*	15.44	89.75	—	—	—
	`FORGE`	**14.31** (**−1.12**)	**92.25** (**+2.50**)	—	—	—

Part B: Periodic 32K Context Benchmark Evaluations Across Steps 229–269 (Pass@1 and Pass@k, %)

		AIME25 (k=8)		AIME26 (k=8)		HMMT_Feb25 (k=8)		HMMT_Feb26 (k=8)		BeyondAIME (k=16)		ArXivMath (k=16)		Micro Average	
Eval Step	**Selector Variant**	Pass@1	Pass@8	Pass@1	Pass@8	Pass@1	Pass@8	Pass@1	Pass@8	Pass@1	Pass@16	Pass@1	Pass@16	**Pass@1**	**Pass@k**
Step 229	*w/o ss*	65.83	86.67	70.42	90.00	51.25	80.00	47.35	69.70	39.69	67.00	22.19	50.00	45.79	71.10
	`FORGE`	**66.25**	**90.00**	68.33	90.00	50.00	**83.33**	46.97	**72.73**	**41.69**	65.00	21.88	50.00	**46.13**	**71.48**
Step 239	*w/o ss*	66.67	86.67	72.08	90.00	50.83	80.00	**53.41**	75.76	**43.25**	67.00	19.53	**55.00**	47.74	72.62
	`FORGE`	**67.08**	86.67	**73.75**	90.00	**53.75**	**83.33**	48.86	**78.79**	42.88	**74.00**	**21.09**	52.50	**47.84**	**75.67**
Step 249	*w/o ss*	**69.58**	86.67	**75.42**	**93.33**	54.58	76.67	47.35	75.76	**45.62**	**75.00**	22.34	**57.50**	**49.45**	**76.05**
	`FORGE`	67.50	86.67	72.08	90.00	**55.00**	**80.00**	**52.27**	**84.85**	45.00	71.00	**22.50**	52.50	49.29	74.90
Step 259	*w/o ss*	68.33	86.67	75.42	93.33	**57.50**	76.67	52.65	78.79	**46.00**	69.00	21.09	55.00	50.26	73.76
	`FORGE`	**70.00**	86.67	**75.83**	93.33	56.67	**83.33**	**53.41**	**81.82**	45.00	**76.00**	**24.69**	**57.50**	**50.67**	**77.95**
Step 269	*w/o ss*	**73.33**	86.67	73.33	90.00	53.33	73.33	**56.82**	78.79	46.31	75.00	22.50	50.00	50.97	74.52
	`FORGE`	72.92	86.67	**78.33**	**93.33**	**60.00**	**83.33**	53.79	78.79	**47.38**	75.00	**22.97**	**52.50**	**52.35**	**76.43**

Table 14. Problem-level mastery and test-time scaling for the controlled 350-step RL comparison across all 312 benchmark problems (9,984 unassisted rollouts per method). Part A counts problems by the number of correct rollouts out of 32 and reports paired solved counts of each baseline against FORGE on identical problems. Part B gives the unbiased micro-average Pass@k curves ($k \in \{1, 2, 4, 8, 16, 32\}$) and the micro-average Avg@32 completed within each generation budget from 8K to 80K tokens.

Part A: Problem-Level Mastery Distribution Out of 312 Problems & Head-to-Head Unlocking vs. FORGE

	Problem Count by Correct Rollouts Out of 32 (%)						Paired Solved (>0/32) vs. FORGE		
Method (350 Steps)	**Unsolved (0)**	**Low (1–4)**	**Mid (5–16)**	**High (17–27)**	**Near-Perf (28–31)**	**Perfect (32)**	**Only FORGE**	**Only Base**	**Net Solved**
GRPO	100 (32.1%)	57 (18.3%)	41 (13.1%)	47 (15.1%)	40 (12.8%)	27 (8.7%)	52	6	+**46**
DAPO	105 (33.7%)	48 (15.4%)	43 (13.8%)	45 (14.4%)	31 (9.9%)	40 (12.8%)	56	5	+**51**
HATCH	102 (32.7%)	48 (15.4%)	46 (14.7%)	38 (12.2%)	40 (12.8%)	38 (12.2%)	53	5	+**48**
SAGE	70 (22.4%)	38 (12.2%)	48 (15.4%)	41 (13.1%)	**61** (**19.6**%)	54 (17.3%)	21	5	+**16**
w/o Hinter RL ($\alpha = 0$)	63 (20.2%)	44 (14.1%)	43 (13.8%)	**59** (**18.9**%)	46 (14.7%)	**57** (**18.3**%)	15	6	+**9**
FORGE ($\alpha = 0.3$)	**54** (**17.3**%)	48 (15.4%)	48 (15.4%)	54 (17.3%)	52 (16.7%)	56 (17.9%)	—	—	—

Part B: Unbiased Micro-Average Pass@k Scaling and Generation-Budget Scaling (Avg@32 Completed $\leq B$ Tokens, %)

	Unbiased Micro Pass@k (%)						Micro Avg@32 Completed Within Budget B (%)							Length & Trunc.	
Method	k=1	k=2	k=4	k=8	k=16	k=32	8K	16K	24K	32K	48K	64K	80K	**Mean Tok.**	**Trunc.**
GRPO	36.48	44.15	50.79	56.96	62.83	67.95	**15.21**	31.39	35.76	36.39	36.48	36.48	36.48	14,221	0.33%
DAPO	37.65	45.08	51.31	56.79	61.88	66.35	13.23	31.02	36.45	37.38	37.65	37.65	37.65	15,270	0.17%
HATCH	37.79	45.38	52.15	58.27	63.42	67.31	14.16	**32.94**	37.37	37.77	37.79	37.79	37.79	13,814	0.07%
SAGE	50.39	58.28	64.53	69.88	74.34	77.56	12.79	29.69	41.59	47.75	50.21	50.38	50.39	21,546	0.42%
FORGE	**51.60**	**60.18**	**67.02**	**73.00**	**78.38**	**82.69**	12.66	29.99	**43.49**	**48.91**	**51.27**	**51.53**	**51.60**	20,723	0.31%

Table 15. Extended test-time Pass@k exploration scaling from $k = 1$ to $k = 512$ across all seven mathematical benchmarks (312 problems in total) under an 81,920-token generation budget, corresponding to Figure 8. We report Pass@k in percent alongside solved problem counts at $k = 32$ and $k = 512$. **Unsaturated Hard-4** pools the four Olympiad and frontier research benchmarks that remain unsaturated at $k = 32$ (**HMMT_Feb26**, **BeyondAIME**, **ArXivMath_0526**, and **ArXivMath_0626**; 222 problems), and **Micro Average** weights all 312 problems equally. Best results within each benchmark comparison are highlighted in **bold**, and rows of FORGE are shaded.

Benchmark (N)	**Model**	Standard Evaluation Horizon (Pass@k, %)						Extended Exploration Horizon (Pass@k, %)				Solved Count	
		k=1	k=2	k=4	k=8	k=16	k=32	k=64	k=128	k=256	k=512	@32	@512
AIME25 (30)	Qwen3.5-4B-Base	81.35	87.27	89.92	92.47	95.32	**96.67**	**96.67**	**96.67**	**96.67**	**96.67**	**29**/30	**29**/30
	FORGE	**86.77**	**90.72**	**92.24**	**93.95**	**95.91**	**96.67**	**96.67**	**96.67**	**96.67**	**96.67**	**29**/30	**29**/30
AIME26 (30)	Qwen3.5-4B-Base	82.50	90.79	93.92	95.28	96.29	**96.67**	**96.67**	**96.67**	**96.67**	**96.67**	**29**/30	**29**/30
	FORGE	**89.48**	**93.64**	**95.02**	**95.96**	**96.59**	**96.67**	**96.67**	**96.67**	**96.67**	**96.67**	**29**/30	**29**/30
HMMT_Feb25 (30)	Qwen3.5-4B-Base	74.27	84.13	**90.00**	**94.84**	**98.30**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**30**/30	**30**/30
	FORGE	**78.33**	**85.31**	88.87	90.49	91.66	93.33	93.33	93.33	96.67	96.67	28/30	29/30
HMMT_Feb26 (33)	Qwen3.5-4B-Base	68.47	77.16	81.48	83.69	85.63	87.88	87.88	**90.91**	90.91	93.94	29/33	31/33
	FORGE	**69.70**	**79.69**	**86.07**	**89.26**	**90.69**	**90.91**	**90.91**	**90.91**	**93.94**	**100.00**	**30**/33	**33**/33
BeyondAIME (100)	Qwen3.5-4B-Base	56.66	65.78	71.84	76.17	80.32	85.00	**90.00**	90.00	93.00	97.00	85/100	97/100
	FORGE	**59.59**	**67.75**	**74.55**	**79.80**	**83.87**	**87.00**	89.00	**91.00**	**95.00**	**98.00**	**87**/100	**98**/100
ArXivMath_0526 (40)	Qwen3.5-4B-Base	24.84	33.22	41.23	48.57	55.24	62.50	65.00	65.00	70.00	77.50	25/40	31/40
	FORGE	**34.92**	**43.48**	**51.64**	**58.51**	**63.67**	**67.50**	**70.00**	**72.50**	**75.00**	**80.00**	**27**/40	**32**/40
ArXivMath_0626 (49)	Qwen3.5-4B-Base	10.91	16.56	24.10	33.49	44.35	55.10	**61.22**	63.27	69.39	73.47	27/49	36/49
	FORGE	**18.75**	**25.74**	**33.89**	**43.05**	**52.86**	**61.22**	**61.22**	**65.31**	**71.43**	**75.51**	**30**/49	**37**/49
Unsaturated Hard-4 (222)	Qwen3.5-4B-Base	42.58	50.74	57.22	62.90	68.65	74.77	78.83	79.73	83.33	87.84	166/222	195/222
	FORGE	**47.64**	**55.88**	**63.16**	**69.26**	**74.40**	**78.38**	**79.73**	**81.98**	**86.04**	**90.09**	**174**/222	**200**/222
Micro Average (312)	Qwen3.5-4B-Base	53.20	61.31	67.04	71.93	76.72	81.41	**84.29**	84.94	87.50	90.71	254/312	283/312
	FORGE	**58.37**	**65.69**	**71.49**	**76.24**	**80.26**	**83.33**	**84.29**	**85.90**	**89.10**	**91.99**	**260**/312	**287**/312

C.1 Supplementary Subgoal Quality Analyses

Figure 9 pairs the quality advantage of reference-free extraction before online filtering with the truncation pressure faced by the later leakage judge. Panel (a) complements the candidate-pool time series in Figure 3a with aggregate quality and leakage measurements for the early-stage windows specified in the caption (protocol in Appendix A.4). Reference-free extraction achieves 79.5% constructive candidate subgoals versus 64.1% with answer-anchored extraction (+15.4 pp), and 83.3% versus 70.8% among admitted training groups (+12.5 pp). Candidate-pool leakage at audited steps 139–219 is 0.0% versus 4.2%.

Online leakage-judge protocol. From step 220 onward, the answer-anchored baseline ($w/\ a^\star$) checks extracted hints after deduplication and before generating hint-guided solver rollouts. The current policy serves as the judge: each request contains the original problem q, its reference answer $a^\star$, and the candidate hint, together with one of two separate test instructions. The judge reasons about the hint and returns `yes` or `no`, where `yes` means that the hint exposes the answer under that test.

1. **Solve the hint: recovering the answer by completing the subtask.** The judge checks whether the hint's requested result gives $a^\star$ directly or through trivial arithmetic. This includes explicit answers, original-question restatements, and boundary claims that fix the target. For example, if the original asks for a total of 42, a hint to evaluate 6×7 exposes that total through an auxiliary calculation. Intermediate counts, setups, and smaller analogous cases receive `no` when the target remains undetermined. The judge assesses the hinted task; no separate solver rollout is generated or scored.
2. **Grant its claim: recovering the answer without completing the subtask.** The judge assumes that whatever the hint asks the student to prove, construct, show, or verify is already true, then checks whether $a^\star$ follows immediately or after one trivial arithmetic step. For example, a hint to prove that all valid configurations partition into six classes of seven supplies the total 42 once the partition claim is accepted, even if proving that claim is difficult. Since solver reward checks the final answer, the student could skip that proof and still be rewarded. An intermediate lemma or identity should receive `no` if substantial work remains even after it is granted.

Voting and failure handling. For each candidate, both tests sample up to four reasoning completions at temperature 0.7. The candidate is rejected if either test produces at least two `yes` votes; it then receives no guided solver rollouts, and its hinter trajectory receives negative feedback in joint RL. Each test first casts three votes and requests the fourth only when exactly one of the first three is `yes`: zero positives cannot reach the threshold, while two or more already meet it. A completed response is parsed for the final boxed verdict, with a trailing `yes`/`no` as fallback. Truncated or unparseable responses contribute no positive (`yes`) verdict, so candidates whose verification votes are truncated are retained by default rather than filtered out.

Truncation during training. The per-vote generation cap is 16,384 tokens at steps 220–289, 32,768 at steps 290–349, and 49,152 at steps 350–379. Figure 9b reports observed truncation under this budget schedule, pooling the number of truncated votes over the number actually cast for each test in each 40-step window. Across the four 40-step windows, truncation rises from 0.27% to 1.34% for the solve-the-hint test and from 0.73% to 3.95% for the grant-its-claim test. Over the same windows, mean rollout response length grows from 9.1K to 18.7K tokens: as the policy generates longer reasoning chains, verification increasingly hits the token cap despite the expanded budget schedule, allowing more low-quality hints to slip through the filter.

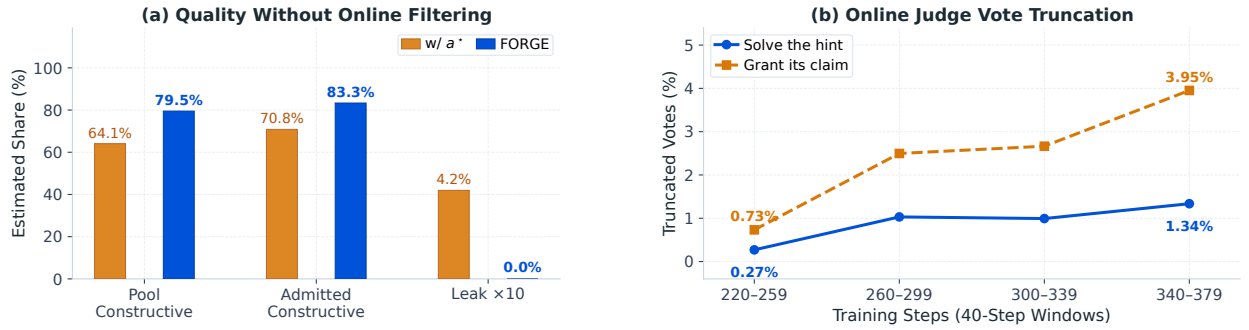

Figure 9. Subgoal quality and online leakage-judge truncation. (a) Before online filtering: candidate-pool Constructive share and shortcut leakage are averaged over audited steps 139, 179, and 219; admitted-group Constructive share is averaged over all six audited steps 19–219. Constructive combines *Useful Subgoal* and *Method / Setup*; shortcut leakage combines *Direct Answer Leak* and *Answer Proxy*. Leakage bars are scaled ×10; annotations show unscaled percentages. **(b)** Online vote truncation for two leakage tests. Each point pools truncated votes over all votes cast for that test within a nonoverlapping 40-step window.

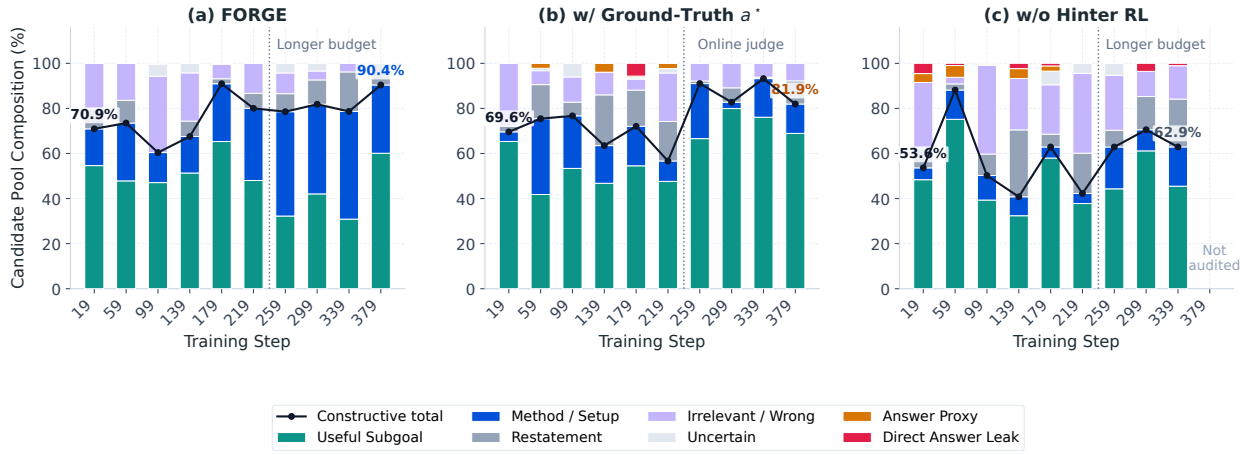

Figure 10. Temporal composition of generated subgoals. Candidate-pool shares across seven mutually exclusive semantic categories for **(a)** `FORGE`, **(b)** answer-anchored joint RL, and **(c)** solver-only updates ($\alpha = 0$), audited every 40 steps from step 19. Bars give inverse-probability-weighted shares; the black boundaries correspond to the Constructive curves in Figure 3a.

Temporal subgoal composition. Figure 10 decomposes the overall Constructive curves in Figure 3a into all seven semantic categories on the same 40-step grid. While online filtering temporarily raises the answer-anchored constructive share over steps 259–339, increasing judge truncation in later training coincides with a rebound in non-constructive outputs (10.3% restatements and 7.8% irrelevant or wrong hints at step 379, reducing its constructive share to 81.9%). In contrast, `FORGE` steadily eliminates irrelevant or wrong extractions (20.2% at step 19 to 0.0% at step 379) and reaches **90.4**% constructive guidance (60.1% explicit useful subgoals and 30.3% method/setup guidance) without an auxiliary verification stage. Table 8 reports the complete per-step estimates and confidence intervals.

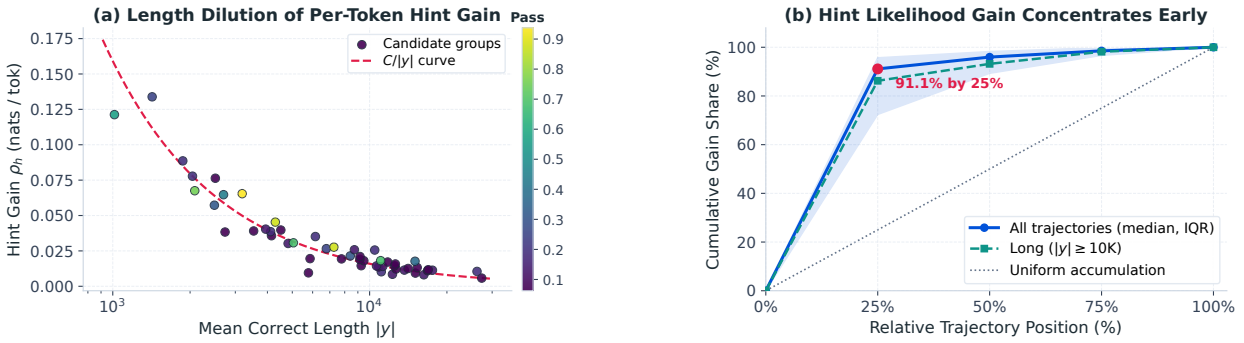

Figure 11. Length dilution of per-token likelihood-ratio transfer weighting in long-horizon reasoning. (a) Length dilution of per-token hint gain: Across 52 mixed groups, mean per-token hinted-vs-unhinted log-likelihood gain ρ_h decays along $C/|y|$ with mean correct response length $|y|$ regardless of guided pass rate (color), so discounting larger ρ_h via w_h (Eq. (12)) favors longer continuations over better hints. **(b) Early prefix concentration:** Across 179 correct trajectories, a median 91.1% of total log-likelihood gain arrives within the first 25% of tokens (even for $|y| \geq 10$K), explaining why longer continuations dilute ρ_h without adding hint gain.

C.2 Length Dilution in Likelihood-Ratio Transfer Weighting

Likelihood-ratio transfer weighting [18, 19] discounts positive hint rewards according to the per-token log-likelihood gain of correct responses $\mathcal{C}_h$ under hinted versus unhinted contexts:

$$S_i = \sum_{t=1}^{|y_i|} \left[\log \pi_\theta(y_{i,t} \mid \tilde{q}, y_{i,<t}) - \log \pi_\theta(y_{i,t} \mid q, y_{i,<t})\right], \qquad \rho_h = \frac{1}{|\mathcal{C}_h|} \sum_{i \in \mathcal{C}_h} \frac{S_i}{|y_i|}, \qquad w_h = \exp\left[-\frac{\max(\rho_h, 0)}{T}\right], \tag{12}$$

where $T > 0$ controls the discount and the positive reward becomes $+\alpha w_h$. In long-horizon mathematical reasoning, however, S_i is heavily concentrated near the beginning of the response: a median **91.1**% of the total gain accumulates in the first 25% of tokens, with negligible accumulation over the remaining 75% (Figure 11b). Because a subgoal primarily steers the initial setup, the solver's own reasoning prefix drives subsequent token generation. Once S_i saturates early, a longer continuation inflates the denominator $|y_i|$ without increasing the numerator, driving ρ_h down along $C/|y|$ both globally (Spearman $r = \mathbf{-0.916}$ with mean correct response length; Figure 11a) and within individual source problems (median within-problem Spearman $r = \mathbf{-1.000}$; Table 12), which systematically inflates w_h. Consequently, per-token length normalization favors longer solver continuations rather than better hints, while shrinking positive rewards to $+\alpha w_h$ (with negative rewards fixed at $-\alpha$) also attenuates the negative advantages of failed extractions (Eq. (7)). Using uniform positive rewards avoids both effects without requiring extra hinted/unhinted likelihood-scoring passes.

C.3 Training Time

Figure 12 compares cumulative GPU hours for the rollout and parameter-update phases across the controlled 350-step runs of Section 4.1, normalized to a matched compute configuration (CP = 2 in the 16K stage and CP = 4 in the 32K stage). In the rollout phase, FORGE incurs comparable GPU hours to DAPO and SAGE while achieving substantially stronger reasoning performance. In the parameter-update phase, because FORGE backfills fixed solver batch slots and optimizes hinter and solver trajectories in a single unified pass, joint hinter–solver training adds minimal overhead: its update cost remains close to GRPO, DAPO, and

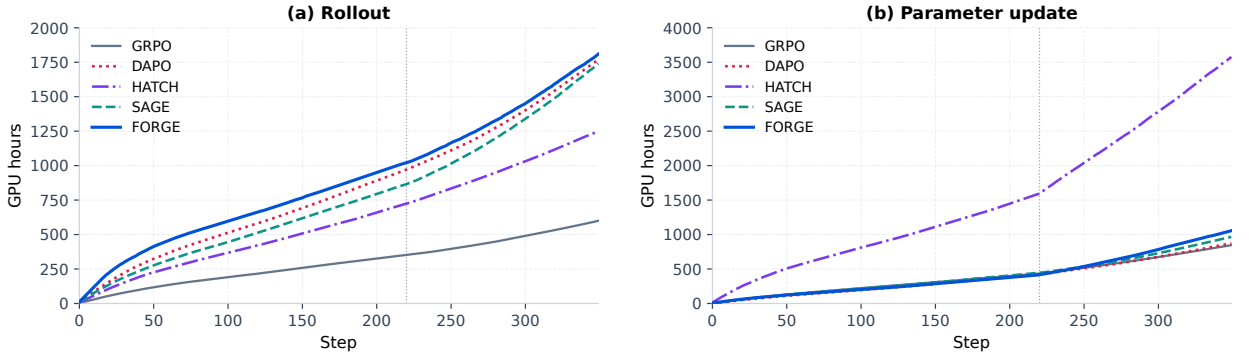

Figure 12. Training time of the controlled 350-step comparison. (a) Cumulative GPU hours of the rollout phase. **(b)** Cumulative GPU hours of the parameter-update phase. All curves are normalized to a matched compute configuration (CP = 2 in the 16K stage and CP = 4 in the 32K stage); the vertical dotted line marks the transition from a 16K to a 32K response budget at step 220.

SAGE, increasing slightly only because `FORGE` generates longer verified-correct reasoning chains with more active gradient tokens per step.

D Why Unlocking all-incorrect Problems Beats Natural Backfilling

This appendix provides the complete derivations and proofs for Theorem 1 in Section 3.5, together with rescue-probability bounds across semantic clusters and an extension to general parameterized policies. Specifically, Proposition 1, Proposition 2, and Corollary 1 below establish Parts (i), (ii), and (iii) of Theorem 1, respectively, explaining why `FORGE` leaves substantially fewer natural all-incorrect groups than DAPO's dynamic sampling [9] despite nearly matched mixed-group batch utilization (Figure 7).

D.1 Three-Mode Shared-Policy Formulation

Consider a shared policy π_z choosing among three reasoning modes $m \in \{a, b, c\}$ via softmax over logits $z = (z_a, z_b, z_c) \in \mathbb{R}^3$, with probabilities $p_j = e^{z_j} / \sum_{u \in \{a,b,c\}} e^{z_u} > 0$:

- Mode a (*erroneous reasoning*) fails both easy and hard problems: $r(q_E, a) = 0$ and $r(q_H, a) = 0$.
- Mode b (*shallow or heuristic reasoning*) solves easy problems but fails hard frontier problems: $r(q_E, b) = 1$ and $r(q_H, b) = 0$.
- Mode c (*deep reasoning*) solves both easy and hard problems: $r(q_E, c) = 1$ and $r(q_H, c) = 1$.

Notice that mode c succeeds on both problem types, so the model introduces no built-in task conflict between q_E and q_H. Let the dataset $\mathcal{D}$ place weight $\pi_E \in (0, 1)$ on easy problems q_E and $\pi_H = 1 - \pi_E \in (0, 1)$ on hard problems q_H. The unassisted success probabilities are $p_E = p_b + p_c = 1 - p_a$ and $p_H = p_c$. Although every individual problem uses a binary verifier $r \in \{0, 1\}$, the true dataset expected reward (unassisted average accuracy) automatically induces a three-level effective reward:

$$J(z) = \pi_E p_E + \pi_H p_H = 0 \cdot p_a + \pi_E \cdot p_b + 1 \cdot p_c = \sum_{j \in \{a,b,c\}} p_j R_j, \qquad R = (0, \pi_E, 1). \tag{13}$$

Using the softmax Jacobian $\partial p_u/\partial z_j = p_u(\delta_{uj} - p_j)$, the exact gradients of $J(z)$ and $p_H(z) = p_c$ are

$$(\nabla_z J)_j = p_j(R_j - J), \qquad (\nabla_z p_c)_j = p_c(\delta_{jc} - p_j). \tag{14}$$

For rollout group size $G \geq 2$, the dataset-wide unassisted all-incorrect rate and its complementary Pass@G coverage are

$$F_G(z) = \pi_E(1-p_E)^G + \pi_H(1-p_H)^G = \pi_E p_a^G + \pi_H(1-p_c)^G, \qquad C_G(z) = 1 - F_G(z). \tag{15}$$

First-step GRPO group gradient. Consider any solver context x whose logits differ from the unassisted logits by a locally constant context shift $\tilde{z} = z + \mathbf{b}_x$, giving sampling probabilities $\tilde{p} = \mathrm{softmax}(\tilde{z})$. Unassisted prompts q_E and q_H have $\mathbf{b}_x = \mathbf{0}$, whereas a subgoal-guided prompt $\tilde{q}_H = \mathcal{P}_{\mathrm{solve}}(q_H, \hat{z})$ shifts the logits so that $\tilde{p}_c > p_c$. At the rollout checkpoint ($\theta = \theta_{\mathrm{old}}$), the importance ratios in Eq. (2) equal one and clipping leaves the first-step surrogate gradient unchanged. Holding the sampled group and advantages fixed during differentiation, we have $\nabla_z \log \tilde{p}_j = e_j - \tilde{p}$, where e_j is the j-th standard basis vector. For any realized binary mixed group of size G with $0 < k < G$ correct responses ($r_i = 1$) and $G - k$ incorrect responses ($r_i = 0$), the group-normalized advantages in Eq. (1) equal $\hat{A}_i = +(G-k)/[G(s_R + \delta)]$ for $r_i = 1$ and $\hat{A}_i = -k/[G(s_R + \delta)]$ for $r_i = 0$, where $s_R = \sqrt{(k/G)(1-k/G)}$. Because $\sum_{i=1}^G \hat{A}_i = 0$, the $\tilde{p}$ term cancels identically and the realized group gradient with respect to z is

$$\hat{g}_x = \frac{1}{G}\sum_{i=1}^{G} \hat{A}_i(e_{o_i} - \tilde{p}) = \lambda_k \left(\frac{1}{k}\sum_{i:r_i=1} e_{o_i} - \frac{1}{G-k}\sum_{i:r_i=0} e_{o_i} \right), \qquad \lambda_k = \frac{k(G-k)}{G^2(s_R+\delta)} > 0. \tag{16}$$

D.2 Why Natural Backfilling (DAPO) Suppresses Unsampled Hard Modes

When a hard problem q_H draws an all-incorrect natural group ($n_c = 0$, occurring with probability $(1-p_c)^G$), its natural GRPO gradient vanishes by Eq. (4). Dynamic sampling in DAPO discards q_H and refills the batch slot with a natural mixed group from an easier problem q_E, where $n_a > 0$ and $n_b + n_c > 0$. Although every such backfilled group has nonzero reward variance, it can still suppress mode c and degrade the true objective.

Proposition 1 (Mode-crowding degradation under natural backfilling; proves Theorem 1(i)). Consider a natural mixed group of size G sampled on q_E with counts (n_a, n_b, n_c) satisfying $n_a > 0$ and $k = n_b + n_c > 0$.

1. *Realized unsampled-winner event ($n_c = 0, n_a > 0, n_b > 0$).* The group gradient is $\hat{g}_{q_E} = \lambda_{n_b}(e_b - e_a)$, and its first-order projections onto p_c and J are

$$\nabla_z p_c^\top \hat{g}_{q_E} = -\lambda_{n_b} p_c(p_b - p_a), \tag{17}$$

$$\nabla_z J^\top \hat{g}_{q_E} = \lambda_{n_b}\big[p_b(\pi_E - J) + p_a J\big] = \lambda_{n_b}\big[\pi_E p_a(2p_b + p_c) - \pi_H p_c(p_b - p_a)\big]. \tag{18}$$

   Hence, whenever $p_b > p_a$, this mixed group strictly decreases the hard-problem pass rate $p_H = p_c$ (raising $(1-p_c)^G$). Whenever $\pi_H p_c(p_b - p_a) > \pi_E p_a(2p_b + p_c)$—which necessarily requires $J > \pi_E$—a sufficiently small gradient ascent step strictly decreases the true dataset expected reward $J(z)$.

2. *Expectation over all mixed groups on q_E.* Conditioned on any success count $0 < k < G$,

$$\nabla_z p_c^\top \mathbb{E}[\hat{g}_{q_E} \mid k] = \frac{\lambda_k p_c}{p_b + p_c}\Big[p_a(p_b + 2p_c) - p_b(p_b - p_c)\Big], \tag{19}$$

$$\nabla_z J^\top \mathbb{E}[\hat{g}_{q_E} \mid k] = \frac{\lambda_k}{p_b + p_c}\Big[\pi_E p_a\big((p_b+p_c)^2 + p_b^2 + p_c^2\big) + \pi_H p_c\big(p_a(p_b + 2p_c) - p_b(p_b - p_c)\big)\Big]. \tag{20}$$

Consequently, whenever $p_b(p_b - p_c) > p_a(p_b + 2p_c)$, natural backfilling strictly shrinks p_c even in expectation over all mixed groups on q_E, and decreases $J(z)$ in expectation whenever the negative second bracketed term outweighs the first.

Proof. For part (1), when $n_c = 0$, all $k = n_b$ successes belong to mode b and all $G - k = n_a$ failures belong to mode a, so Eq. (16) gives $\hat{g}_{q_E} = \lambda_{n_b}(e_b - e_a)$. Dotting with Eq. (14) yields $\nabla_z p_c^\top \hat{g}_{q_E} = \lambda_{n_b}(-p_c p_b + p_c p_a) = -\lambda_{n_b} p_c(p_b - p_a)$ and $\nabla_z J^\top \hat{g}_{q_E} = \lambda_{n_b}[p_b(\pi_E - J) - p_a(0 - J)]$. Substituting $J = \pi_E(p_b + p_c) + \pi_H p_c$ and $1 - p_b - p_c = p_a$ simplifies the bracket to $\pi_E p_b - (p_b - p_a)[\pi_E(p_b + p_c) + \pi_H p_c] = \pi_E[p_b p_a + p_a(p_b + p_c)] - \pi_H p_c(p_b - p_a) = \pi_E p_a(2p_b + p_c) - \pi_H p_c(p_b - p_a)$. Because $p_a J > 0$, negativity of $p_b(\pi_E - J) + p_a J$ implies $J > \pi_E$.

For part (2), independence of rollouts implies that, conditional on $0 < k < G$ successes on q_E, each success equals b with probability $p_b/(p_b + p_c)$ and c with probability $p_c/(p_b + p_c)$, while every failure equals a. Taking expectations in Eq. (16) gives $\mathbb{E}[\hat{g}_{q_E} \mid k] = \lambda_k\big(\frac{p_b}{p_b+p_c} e_b + \frac{p_c}{p_b+p_c} e_c - e_a\big)$. Dotting with $\nabla_z p_c = (-p_c p_a, -p_c p_b, p_c(1 - p_c))$ yields

$$\nabla_z p_c^\top \mathbb{E}[\hat{g}_{q_E} \mid k] = \frac{\lambda_k p_c}{p_b + p_c}\big[p_c(1 - p_c) - p_b^2 + p_a(p_b + p_c)\big] = \frac{\lambda_k p_c}{p_b + p_c}\big[p_a(p_b + 2p_c) - p_b(p_b - p_c)\big], \quad (21)$$

where we used $1 - p_c = p_a + p_b$. Similarly, since $p_E = 1 - p_a$ has gradient $\nabla_z p_E = -\nabla_z p_a = (-p_a(1 - p_a), p_a p_b, p_a p_c)$, we obtain

$$\nabla_z p_E^\top \mathbb{E}[\hat{g}_{q_E} \mid k] = \lambda_k p_a \left[(1 - p_a) + \frac{p_b^2 + p_c^2}{p_b + p_c}\right] = \frac{\lambda_k p_a}{p_b + p_c}\big[(p_b + p_c)^2 + p_b^2 + p_c^2\big]. \quad (22)$$

Combining $\nabla_z J = \pi_E \nabla_z p_E + \pi_H \nabla_z p_c$ gives Eq. (20). □

The mechanism is purely geometric and requires no conflict between q_E and q_H. Because q_E's binary verifier assigns reward 1 equally to shallow mode b and deep mode c, reinforcing q_E directs a fraction $p_b/(p_b + p_c)$ of positive credit to z_b and only $p_c/(p_b + p_c)$ to z_c. When b dominates and c is scarce, raising z_b expands the softmax partition function faster than e^{z_c}, crowding out the deep mode c and locking q_H into repeated all-incorrect outcomes.

D.3 Why Unlocking all-incorrect Problems (FORGE) Reverses the Update Direction

Unlocking q_H requires both sampling mode c and converting the resulting group into a reliable unassisted improvement.

Access to mode c via subgoal guidance and semantic clusters. For a single rollout group of size G with success probability p, the probability of observing a mixed group is $m_G(p) = 1 - (1 - p)^G - p^G$. Spending M additional natural rollout groups retrying q_H at a frozen checkpoint produces at least one mixed group with probability

$$R_{\text{nat}}(M) = 1 - [1 - m_G(p_c)]^M \le MGp_c, \quad (23)$$

which vanishes linearly as $p_c \to 0$. In FORGE, extracting M subgoal candidates $\hat{z}_1, \ldots, \hat{z}_M$ from the failed prefixes produces guided contexts $\tilde{q}_{H,1}, \ldots, \tilde{q}_{H,M}$ with lifted probabilities $\tilde{p}_{c,j} > p_c$. Conditional on these contexts, independent guided groups yield at least one mixed group with probability

$$R_{\text{guide}} = 1 - \prod_{j=1}^{M} [1 - m_G(\tilde{p}_{c,j})]. \quad (24)$$

If a candidate list contains a subgoal with $\tilde{p}_{c,j} \in [\varepsilon, 1-\varepsilon]$ with probability at least γ, the rescue probability remains at least $\gamma m_G(\varepsilon)$ even as $p_c \to 0$. Moreover, if each semantic cluster carries an independent latent mixed-group probability X whereas M repeats within one cluster share the same X, convexity of $x \mapsto (1-x)^M$ implies $1-(1-\mathbb{E}[X])^M \geq 1-\mathbb{E}[(1-X)^M]$ (strict whenever $M > 1$ and X varies), explaining why semantic cluster coverage lifts rescue probability [36]. Under Eq. (6), the expected hinter reward $\mathbb{E}[r^{\text{hint}} \mid \tilde{q}] = \alpha[2m_G(p_{\tilde{q}}) - 1]$ directly trains the hinter to propose contexts that maximize $m_G(p_{\tilde{q}})$.

Proposition 2 (Single-sample rescue guarantees positive improvement; proves Theorem 1(ii)). Suppose subgoal guidance on $\tilde{q}_H$ yields any realized mixed group in which mode c is sampled at least once ($n_c = k \in \{1, \ldots, G-1\}$, $n_a + n_b = G - k \geq 1$). Write $\alpha_a = n_a/(G-k)$ and $\alpha_b = n_b/(G-k)$ ($\alpha_a, \alpha_b \geq 0$, $\alpha_a + \alpha_b = 1$). Then the realized GRPO gradient $\hat{g}_{\tilde{q}_H} = \lambda_k(e_c - \alpha_a e_a - \alpha_b e_b)$ satisfies:

1. *Unconditional hard-problem unlocking:*

$$\nabla_z p_c^\top \hat{g}_{\tilde{q}_H} = \lambda_k p_c(1 - p_c + \alpha_a p_a + \alpha_b p_b) \geq \lambda_k p_c(1 - p_c) > 0. \tag{25}$$

   Every rescued group—even a single $1/G$ breakthrough—strictly increases the unassisted hard-problem pass rate $p_H = p_c$ and strictly lowers $(1-p_c)^G$.

2. *No degradation of true expected reward in the entire mode-crowding regime:*

$$\nabla_z J^\top \hat{g}_{\tilde{q}_H} = \lambda_k \Big[p_c(1-J) + \alpha_a p_a J + \alpha_b p_b (J - \pi_E) \Big]. \tag{26}$$

   Whenever $J \geq \pi_E$—which strictly contains every state where natural backfilling can decrease J in Proposition 1—all three terms are nonnegative and the first is strictly positive, yielding

$$\nabla_z J^\top \hat{g}_{\tilde{q}_H} \geq \lambda_k p_c(1-J) > 0 \tag{27}$$

   for every realized mixed group with $k \geq 1$ and every failure composition (α_a, α_b).

Proof. On $\tilde{q}_H$, the verifier grades against the original hard problem q_H, so $r(q_H, c) = 1$ and $r(q_H, a) = r(q_H, b) = 0$. Every correct response is therefore mode c, and the failures consist of n_a draws of a and n_b draws of b. Substituting into Eq. (16) gives $\hat{g}_{\tilde{q}_H} = \lambda_k(e_c - \alpha_a e_a - \alpha_b e_b)$. Dotting with $\nabla_z p_c = (-p_c p_a, -p_c p_b, p_c(1-p_c))$ immediately gives Eq. (25), and since $\alpha_a p_a + \alpha_b p_b > 0$, the lower bound $\lambda_k p_c(1-p_c) > 0$ holds unconditionally. Dotting with $\nabla_z J = (-p_a J, p_b(\pi_E - J), p_c(1-J))$ gives

$$\nabla_z J^\top \hat{g}_{\tilde{q}_H} = \lambda_k \big[p_c(1-J) - \alpha_a(-p_a J) - \alpha_b p_b(\pi_E - J) \big] = \lambda_k \big[p_c(1-J) + \alpha_a p_a J + \alpha_b p_b (J - \pi_E) \big]. \tag{28}$$

Since $0 < J < 1$, the first term is strictly positive, the second is nonnegative, and when $J \geq \pi_E$ the third is also nonnegative, proving Eq. (27). □

The contrast with Proposition 1 is sharp. On q_E, binary verification cannot distinguish b from c, so b receives positive advantage and crowds out c. On the unlocked hard problem $\tilde{q}_H$, binary verification assigns $r = 0$ (negative advantage) to b and $r = 1$ (positive advantage) exclusively to c; sampling c even once therefore flips the update direction on both p_c and J.

Corollary 1 (Dataset-wide all-incorrect rate F_G and Pass@G coverage; proves Theorem 1(iii)). From Eq. (15), the gradient of dataset-wide Pass@G coverage $C_G = 1 - F_G$ is

$$\nabla_z C_G = -\nabla_z F_G = G \Big[\pi_E p_a^{G-1} \nabla_z p_E + \pi_H (1-p_c)^{G-1} \nabla_z p_c \Big] = S_G \, \nabla_z \tilde{J}, \tag{29}$$

where $S_G = G[\pi_E p_a^{G-1} + \pi_H(1-p_c)^{G-1}] > 0$ and $\tilde{J} = \tilde{\pi}_E p_E + \tilde{\pi}_H p_c$ has effective weights $\tilde{\pi}_E = \pi_E p_a^{G-1}/[\pi_E p_a^{G-1} + \pi_H(1-p_c)^{G-1}]$ and $\tilde{\pi}_H = 1 - \tilde{\pi}_E$. Because $p_a = 1 - p_c - p_b < 1 - p_c$ whenever $p_b > 0$, the weight ratio $\tilde{\pi}_E/\tilde{\pi}_H = (\pi_E/\pi_H)[p_a/(1-p_c)]^{G-1}$ decays exponentially in G. Consequently, whenever $\pi_H p_c(1-p_c)^{G-1} \geq \pi_E p_a^G$—a condition satisfied by orders of magnitude at $G = 16$ even when hard problems are a small minority—we automatically have $\tilde{J} \geq \tilde{\pi}_E$:

- By Proposition 2 applied to $\tilde{J}$, *every* rescued FORGE group ($k \geq 1$) strictly lowers the dataset-wide unassisted all-incorrect rate F_G ($\nabla_z F_G^\top \hat{g}_{\tilde{q}_H} < 0$).
- By Proposition 1 applied to $\tilde{J}$, whenever $p_b(p_b - p_c) > p_a(p_b + 2p_c)$ and G is sufficiently large, DAPO's natural backfilling on q_E strictly *increases* F_G both on the $n_c = 0$ event and in expectation over all mixed groups.

Coupled fixed-batch comparison. In a coupled one-step update from a shared checkpoint where both methods retain the identical natural mixed groups and fill every remaining slot to full capacity, DAPO backfills with q_E whereas FORGE backfills with a rescued $\tilde{q}_H$. Under the conditions of Corollary 1, every backfilled slot satisfies $\nabla_z p_c^\top(\hat{g}_{\tilde{q}_H} - \mathbb{E}[\hat{g}_{q_E} \mid k]) > 0$ and $\nabla_z F_G^\top(\hat{g}_{\tilde{q}_H} - \mathbb{E}[\hat{g}_{q_E} \mid k]) < 0$, giving FORGE a strictly stronger first-order reduction in the unassisted all-incorrect rate at identical mixed-group batch utilization.

D.4 Extension to General Parameterized Policies

In a general autoregressive policy π_θ with constant response length L, let $p_x = \Pr_{o\sim\pi_\theta(\cdot|x)}[r^{\text{solve}}(q,o) = 1]$ and $\ell_x = \log[p_x/(1-p_x)]$ for $0 < p_x < 1$. By the score identity $\mathbb{E}[\nabla_\theta \log \pi_\theta(o \mid x) \mid r = 1] = \nabla_\theta p_x / p_x$ and $\mathbb{E}[\nabla_\theta \log \pi_\theta(o \mid x) \mid r = 0] = -\nabla_\theta p_x/(1-p_x)$, the token-averaged group gradient at the rollout checkpoint satisfies the count-conditional log-odds identity

$$\mathbb{E}[\hat{g}_x \mid x, k] = \frac{\lambda_k}{L} \nabla_\theta \ell_x \tag{30}$$

for any mixed group with $0 < k < G$ (cf. binary-reward gradient alignment in GradAlign, Theorem 4.2 and Appendix C.2).[2] Applying Eq. (30) to unassisted Pass@K coverage $C_K(q;\theta) = 1 - (1-p_q)^K$, whose gradient is $\nabla_\theta C_K(q;\theta) = K p_q (1-p_q)^K \nabla_\theta \ell_q$, the first-order expected transfer from a rescued guided group on $\tilde{q}$ is

$$\mathbb{E}[\Delta C_K(q) \mid \tilde{q}, k] = \eta \frac{\lambda_k}{L} K p_q (1-p_q)^K \langle \nabla_\theta \ell_q, \nabla_\theta \ell_{\tilde{q}} \rangle + O(\eta^2). \tag{31}$$

In the three-mode softmax model of Sections D.1–D.3, any additive subgoal shift $\tilde{z} = z + \mathbf{b}$ automatically guarantees positive alignment $\langle \nabla_z \ell_{q_H}, \nabla_z \ell_{\tilde{q}_H} \rangle = 1 + \frac{p_a \tilde{p}_a + p_b \tilde{p}_b}{(p_a+p_b)(\tilde{p}_a+\tilde{p}_b)} > 1$ (with exact equality $\nabla_z \ell_{\tilde{q}_H} = \nabla_z \ell_{q_H}$ when $b_a = b_b$). In a full neural policy, Eq. (31) shows that unlocking all-incorrect problems transfers to unassisted coverage whenever subgoal guidance raises $\tilde{p}_q$ while maintaining positive gradient alignment $\langle \nabla_\theta \ell_q, \nabla_\theta \ell_{\tilde{q}} \rangle > 0$—a condition that FORGE protects by using reference-free, question-only subgoals rather than answer-leaking shortcuts (Section 3.1). This theoretical contrast aligns with the training dynamics in Figure 7: although both DAPO and FORGE fill nearly all solver batch slots with mixed-outcome groups, FORGE drives the unassisted all-incorrect ratio markedly lower.

E Case Study

[2] https://arxiv.org/abs/2602.21492

E.1 Multi-Entry Subgoal Complementarity vs. Single-Path Hinting

To concretely illustrate why extracting and grouping multiple subgoals from diverse failed rollouts (Sections 3.1–3.2 and Figure 3b–c) rescues frontier problems that single-path reference-solution hinting misses, we present a side-by-side comparison between SAGE [20] and FORGE on the exact same training problem encountered at step 309.

Source Frontier Problem at Step 309 (Natural Rollouts: 0/16 Correct, 11/16 Truncated)

Problem (q): How many ways can a 4×4 board be filled with nonnegative integers such that the sum of the numbers in each row and each column is 3?

Ground-Truth Answer ($a^\star$): 2008

Single-path reference-solution hinting in SAGE (0/16 correct). When the initial 16 unhinted rollouts fail, SAGE escalates to its maximum level-3 hint derived from the reference solution and broadcasts a single shared hint across all 16 guided solver rollouts:

> *"This problem is about counting integer solutions to a system of linear equations. Think about using the stars and bars method to first count the row configurations, then consider how the column sums constrain the choices for subsequent rows. Is there a known combinatorial formula or principle that applies to this specific case of a 4x4 board with row and column sums of 3?"*

Because naively applying stars-and-bars row by row leads to severe branching over asymmetric column-sum residuals across four rows, all 16 guided rollouts in SAGE get bogged down in casework or miscount overlaps, finishing with 0/16 correct solutions.

Multi-entry failure-grounded subgoal exploration in FORGE (2/16 unlocked via a unique cluster). From the 16 failed natural thinking prefixes (which variously attempted Birkhoff–von Neumann permutation decomposition, single-row weak compositions, and bounded-entry casework), FORGE extracts 11 valid subgoal groups and clusters them via $\mathcal{P}_{\text{dedup}}$ into **9 distinct semantic clusters** (including 2 alternative formulations retained within the candidate budget; Appendix A.1). As shown in Table 16, eight of the nine semantic directions—including binary $\{0,1\}$ matrix reduction (Cluster 0), single-row stars-and-bars (Cluster 1, mirroring SAGE's direction), permutation triples (Cluster 2), bounded $\{0,1,2\}$ entries (Clusters 7 and 8), bipartite matchings (Cluster 12), and the still-complex 3×3 reduction (Cluster 14)—fail completely (0/16 or 0/32 across 160 guided trajectories), confirming that additional rollouts along unproductive formulations do not unlock the problem.

Crucially, **Cluster 6** asks the solver to first count 2×2 nonnegative integer boards with row and column sums equal to 3. Conditioned on this single $G = 16$ subgoal group, 2/16 solver trajectories successfully unlock the original 4×4 problem: after rapidly solving the 2×2 warmup (4 boards, parameterized by a single free entry $a_{11} \in \{0,1,2,3\}$), the solver generalizes the two-row structure by partitioning the 4×4 board into top and bottom 2×4 halves indexed by the column-sum profile $\mathbf{c} = (c_1, c_2, c_3, c_4)$ of the first two rows ($\sum_{j=1}^{4} c_j = 6$, $0 \le c_j \le 3$). By symmetry between the top two rows (column sums c_j) and bottom two rows (column sums $3 - c_j$), the solver computes the number of valid two-row configurations $N(\mathbf{c})$ for each partition type of 6 and sums $N(\mathbf{c}) \cdot N(\mathbf{3} - \mathbf{c})$ across all five profile classes— obtaining $96 + 864 + 224 + 224 + 600 =$ **2008**—converting a 0/16 dead end into a verified mixed training group ($p_{\text{guide}} = 2/16$).

Table 16. All 9 distinct semantic subgoal clusters (11 evaluated subgoal groups) extracted by FORGE from the $G = 16$ failed natural rollouts on the 4×4 contingency-table problem at step 309 ($a^\star = 2008$). Only Cluster 6 unlocks a learnable mixed solver group (2/16).

Cluster ID	Cand. ID	Extracted Question-Only Subgoal ($\hat{z}_{q,k}$)	Role	Guided Pass Rate
0	0	How many 4x4 matrices with entries 0 or 1 have each row and column sum equal to 3?	Representative	0/16
0	4	How many 4×4 matrices with entries 0 or 1 satisfy that each row sum is 3 and each column sum is 3?	Alternative	0/16
1	1	How many ways can a single row of a 4x4 board be filled with nonnegative integers such that the sum of the numbers is 3?	Representative	0/16
1	10	What is the number of ways to distribute 3 indistinguishable balls into 4 distinguishable boxes?	Alternative	0/16
2	2	What is the number of unordered triples of permutations of the set {1,2,3,4}?	Representative	0/16
3	3	How many ways can a 2×2 board be filled with nonnegative integers such that the sum of the numbers in each row and each column is 2?	Representative	0/16
6	**6**	How many ways can a 2×2 board be filled with nonnegative integers such that the sum of the numbers in each row and each column is 3?	**Representative**	**2/16 (Unlocked)**
7	7	How many 4×4 matrices with entries 0, 1, or 2 satisfy that each row and each column sum to 3?	Representative	0/16
8	8	How many 3 × 3 matrices with entries from {0, 1, 2} have every row and every column sum equal to 3?	Representative	0/16
12	12	How many perfect matchings are there in the complete bipartite graph $K_{4,4}$?	Representative	0/16
14	14	How many ways can a 3×3 board be filled with nonnegative integers such that the sum of the numbers in each row and each column is 3?	Representative	0/16

E.2 Reference-Free Subgoals vs. Reference- and Answer-Anchored Spoilers

To illustrate why conditioning hint generation on reference solutions or the ground-truth answer $a^\star$ induces shortcut leakage and superficial restatements (Section 3.1, Figures 1 and 3a), we present three representative cases from our source-blinded semantic audit.

Case 2A: Reference-solution hinting discloses decisive steps as arithmetic proxies. When an unhinted group fails in SAGE, escalating along a privileged reference solution frequently hands the solver the decisive mathematical construction, reducing an Olympiad- or research-level proof to trivial arithmetic while still producing a mixed training group ($0 < p_{\text{guide}} < 1$).

Case 2A: Reference-Based Decisive-Step Spoilers in Admitted Mixed Groups (SAGE)

Example 1 (Board Domination, Step 59, Level 2, $a^\star = 50$): What is the smallest k such that k cells can be marked on a 10×11 board so that any placement of a three-cell L-shaped piece will touch at least one marked cell?

- **SAGE Hint (answer_proxy, $p_{\text{guide}} = 14/16$):** *"Using the condition from the previous hint, consider the 10x10 sub-board. Partition it into disjoint 2x2 blocks. How many such blocks are there, and what is the minimum number of marked cells needed if each block must contain at least two marked cells?"*
- **Audit Analysis:** Partitioning the 10×10 sub-board into 25 disjoint 2×2 blocks with 2 marked cells per block reduces the problem to computing $25 \times 2 = 50 = a^\star$, bypassing both the discovery of the sub-board partition and the upper-bound construction.

Example 2 (Lie Algebra Homomorphisms, Step 219, Level 3, $a^\star = 7$): How many injective homomorphisms are there between the Lie algebras $\mathfrak{sl}_2$ and $\mathfrak{sp}_6$, up to conjugation by Sp_6?

- **SAGE Hint (`answer_proxy`, $p_{\text{guide}} = 14/16$):** *"Examine the valid partitions of 6 where odd parts have even multiplicities. Exclude the partition corresponding to the zero orbit (since we need injective homomorphisms), and count the remaining valid partitions."*
- **Audit Analysis:** The hint supplies the complete classification theorem translating $\mathfrak{sl}_2 \hookrightarrow \mathfrak{sp}_6$ conjugacy classes into symplectic partitions of 6, leaving only mechanical enumeration ($8 - 1 = 7 = a^\star$).

Case 2B: Answer-anchored extraction vs. reference-free decomposition on identical problems. Even without a full reference solution, exposing the scalar answer $a^\star$ to the hinter allows it to work backward from the target value—either embedding $a^\star$ into a quantitative assertion to prove or restating the original target. Withholding $a^\star$ forces `FORGE` to extract genuine intermediate subproblems from the failed reasoning prefixes.

Case 2B: Side-by-Side Comparison on Identical Frontier Problems at Step 219

Problem 1 (Circular Inequality Extremal Count, $a^\star = 49$): Around a circle, 100 integers are written. Each of these numbers is greater than the sum of the two numbers that follow it in a clockwise direction. What is the maximum number of positive numbers that can be among the written numbers?

- **w/ $a^\star$ (`answer_proxy`, $p_{\text{guide}} = 16/16$, zero GRPO variance):** `Prove that there are at least 51 negative integers among the 100 written numbers. This subgoal can guide the solver toward an upper bound on the number of positive integers, as minimizing the number of negatives implies maximizing the positives. After establishing that at least 51 numbers must be negative, the maximum possible number of positive integers follows directly.`
- **w/ $a^\star$ (`restatement`, $p_{\text{guide}} = 1/16$):** `What is the maximum possible number of positive integers that can appear among the 100 integers written around the circle while satisfying the condition` $a_i > a_{i+1} + a_{i+2}$ `for every index` i`?`
- `FORGE` (`useful_subgoal`, $p_{\text{guide}} = 1/16$, **unlocked**): `What constraints must be satisfied if two consecutive integers are positive?`

Mechanism: Conditioning on $a^\star = 49$ lets the hinter encode $100 - 49 = 51$ directly into the prompt (collapsing all 16 solver rollouts to 49 with zero gradient signal) or fall back to a verbatim restatement. Without $a^\star$, `FORGE` isolates the key structural lemma—that two adjacent positive entries propagate counterclockwise around the cycle to force all entries positive, yielding a contradiction—while leaving the solver to rule out the alternating 50-positive case and construct a valid 49-positive configuration.

Problem 2 (Induced Subgraphs in Acyclic Graphs, $a^\star = 1517$): In every acyclic graph with 2022 vertices we can choose k of the vertices such that every chosen vertex has at most 2 edges to chosen vertices. Find the maximum possible value of k.

- **w/ $a^\star$ (`answer_proxy`, $p_{\text{guide}} = 16/16$, zero GRPO variance):** `Show that in any acyclic graph with` n `vertices, there exists a set of at least` $\lceil \frac{3}{4}n \rceil$ `vertices such that each chosen vertex has at most 2 neighbors among the chosen vertices.`
- `FORGE` (`useful_subgoal`, $p_{\text{guide}} = 1/16$, **unlocked**): `In a forest where every internal vertex has exactly two pendant leaves, what is the maximum number of vertices that can be chosen so that each chosen vertex has at most two edges to vertices in the chosen set?`

Mechanism: The answer-anchored hint supplies the exact $\lceil 3n/4 \rceil$ formula (evaluating immediately to $\lceil 6066/4 \rceil = 1517$), whereas `FORGE` proposes a concrete extremal tree family from the failed prefix's structural exploration without disclosing the target bound.

Case 2C: How midway-impasse prefixes directly reveal intermediate subgoals. Our prefix audit (Figure 1b; Table 6) shows that 71.6% of 0/16 rollouts establish a valid setup before reaching a midway reasoning impasse at a specific intermediate obstacle. Below we trace how `FORGE` turns two distinct midway impasses on the same Bessel-function integral into complementary subproblems that unlock the group without $a^\star$.

Case 2C: From Midway Reasoning Impasses to Unlocking Subgoals (Step 19)

Problem (q): Calculate the integral $\int_{-\infty}^{\infty} \frac{J_1(x-a)}{x-a} \frac{J_1(x-b)}{x-b}\, dx$. **Ground-Truth Answer ($a^\star$):** $\frac{8(\sin(a-b)-(a-b)\cos(a-b))}{\pi(a-b)^3}$

- **Failed Prefix 1 (midway impasse, truncated at 16,384 tokens):** Correctly reduces the integral to a shift-invariant convolution in $d = a - b$ and invokes Plancherel's theorem, but reaches an impasse while attempting to evaluate the Fourier transform $F(k) = \int_{-\infty}^{\infty} \frac{J_1(x)}{x} e^{-ikx}\, dx$ in closed form.
- **Extracted Subgoal 1 (`useful_subgoal`, $p_{\text{guide}} = 4/16$, unlocked):** `Compute the Fourier transform of the function` $f(x) = \frac{J_1(x)}{x}$, `i.e., find` $F(k) = \int_{-\infty}^{\infty} \frac{J_1(x)}{x} e^{-ikx}\, dx$.
- **Failed Prefix 2 (midway impasse, truncated at 16,384 tokens):** Attempts a series and Parseval reduction on the same integral but reaches an impasse calibrating the $a = b$ normalization constant before handling the general shift $a - b$.
- **Extracted Subgoal 2 (`useful_subgoal`, $p_{\text{guide}} = 5/16$, unlocked):** `Evaluate the integral` $\int_{-\infty}^{\infty} \left(\frac{J_1(x)}{x}\right)^2 dx$.

Mechanism: Neither failed rollout reaches the final closed form, yet each prefix walks directly up to a well-defined analytical bottleneck; isolating that bottleneck as a standalone subproblem lets the guided solver resolve the obstacle first and complete the convolution integral.

E.3 Co-Evolution of Problem Decomposition via Direct Hinter RL

Section 3.4 and Figure 3a show that freezing the hinter to the evolving solver weights without direct hinter RL ($\alpha = 0$) leaves a large fraction of extractions as trivial restatements or off-target questions. Below we compare extractions from *w/o Hinter RL* ($\alpha = 0$) and `FORGE` ($\alpha = 0.3$) on the same architectural geometry problem.

Case 3: Direct Hinter RL ($\alpha = 0.3$) vs. Solver-Only Updates ($\alpha = 0$) on Geometric Decomposition

Problem (q, $a^\star = 100$): The keystone arch is an ancient architectural feature. It is composed of congruent isosceles trapezoids fitted together along the non-parallel sides. The bottom sides of the two end trapezoids are horizontal. In an arch made with 9 trapezoids, let x be the angle measure in degrees of the larger interior angle of the trapezoid. Find the value of x.

Extractions without Hinter RL ($\alpha = 0$, Step 139):

- **Candidate (`restatement`, $p_{\text{guide}} = 0/16$):** `In a symmetrical keystone arch composed of 9 congruent isosceles trapezoids, with the bottom sides of the two end trapezoids horizontal, what is the number of trapezoids on one side of the arch?`
- **Audit Analysis:** Asking for $(9-1)/2 = 4$ merely rephrases a given count from the problem statement and provides no geometric link between the trapezoid leg angles and the arch curvature, leaving the group all-incorrect ($0/16$).

Extractions with Direct Hinter RL in `FORGE` ($\alpha = 0.3$, Steps 139 & 339):

- **Step 139 Candidate 1 (`useful_subgoal`, $p_{\text{guide}} = 1/16$, unlocked):** `What is the total angular change required from the vertical axis to the horizontal base of the arch on one side?`
- **Step 139 Candidate 2 (`useful_subgoal`, $p_{\text{guide}} = 1/16$, unlocked):** `What is the sum of the smaller interior angles of the trapezoids that meet at the keystone if there are 9 trapezoids?`
- **Step 339 Candidate (`method_setup`, $p_{\text{guide}} = 6/16$, unlocked):** `How many incremental changes of 2x - 180 degrees occur between consecutive trapezoids, and what total rotation must this sum equal for the arch to close properly?`

Mechanism: Direct outcome-based hinter rewards penalize $0/16$ restatements ($-\alpha$) and reinforce questions that isolate the angular deflection across shared non-parallel legs ($+\alpha$), progressively sharpening the hinter's geometric decomposition from step 139 to step 339.

Table 17. Representative comparison between low-pass-rate ($p_{\text{guide}} = 1/16$) frontier breakthroughs and high-pass-rate ($p_{\text{guide}} \geq 12/16$) extractions in the 32K stage (steps 220–270). Low p_{guide} groups frequently reflect crisp intermediate lemmas on multi-step frontier problems, whereas high p_{guide} groups often arise on near-threshold problems via superficial clarifications or answer proxies.

Regime	Step / p_{guide}	Source Problem Summary ($a^\star$)	Extracted Subgoal ($\hat{z}_{q,k}$)	Audit Category	$4p(1-p)$	FORGE
Frontier Breakthrough (1/16)	Step 232 1/16	Evaluate $\int_0^\infty \frac{\log(1+x)}{(1+x^2)(1+x^3)}\,dx$ ($a^\star = \frac{G}{2} + \frac{\pi}{8}\log 2 - \frac{37\pi^2}{864}$)	`What is the partial fraction decomposition of` $\frac{1}{(1+x^2)(1+x^3)}$?	`useful_subgoal`(Score 3/3)	$+0.23\alpha$	$+\alpha$
	Step 263 1/16	Smallest $\lvert A\rvert$ s.t. $\forall m$, $\exists a \in A, n \geq 1$ with $a^n \equiv m \pmod{100}$ ($a^\star = 41$)	`What residues modulo 4 can be obtained as powers of an integer, and what residues modulo 25 can be obtained as powers of a unit?`	`useful_subgoal`(Score 3/3)	$+0.23\alpha$	$+\alpha$
	Step 269 1/16	Pairwise GCDs of 5 integers are $2,3,4,5,6,7,8,p,q,r$; find $\min(p+q+r)$ ($a^\star = 9$)	`How can we distribute the exponents of 2 among the five integers to satisfy the GCD conditions for 2, 4, 6, and 8?`	`useful_subgoal`(Score 3/3)	$+0.23\alpha$	$+\alpha$
High Pass Rate (13–14/16)	Step 259 14/16	Probability that a random arrangement of "HMMT2005" reads "HMMT" or "2005" left to right ($a^\star = 23/144$)	`Does "reading" "HMMT" from left to right require the letters to be consecutive, or can there be other characters in between?`	`restatement`(Score 2/3)	$+0.44\alpha$	$+\alpha$
	Step 267 14/16	Heaviest weight obtainable using at most two weights from 2, 4, 12 lb ($a^\star = 16$)	`What is the sum of the two heaviest single weights?`	`answer_proxy`(Score 1/3)	$+0.44\alpha$	$+\alpha$

E.4 Why 1/16 Frontier Breakthroughs Warrant Full Hinter Credit

Section 4.3 and Figure 6c establish that guided pass rate p_{guide} reflects the residual difficulty of completing the original problem rather than the pedagogical quality of the subgoal. Table 17 contrasts three $p_{\text{guide}} = 1/16$ frontier breakthroughs—which midpoint-favoring shaping $4p_{\text{guide}}(1 - p_{\text{guide}})$ penalizes with a 4.3× reward discount (0.234α vs. 1.0α)—against high-pass-rate (12/16–15/16) extractions from the same 32K training window (steps 220–270).

As Table 17 illustrates, on the Catalan-constant integral (Step 232), the modular power-covering problem (Step 263), and the pairwise-GCD extremal problem (Step 269), the hinter formulates textbook mathematical decompositions—partial fractions, Chinese Remainder Theorem splitting modulo 4 and 25, and 2-adic valuation assignment. Because completing the remaining integration or extremal casework still challenges the solver, only 1/16 rollouts reach the final answer, yet that single verified trajectory restores a non-zero GRPO learning signal on a previously unsolvable problem. By assigning a uniform $+\alpha$ across all mixed groups ($0 < p_{\text{guide}} < 1$), FORGE gives full credit to these high-value frontier breakthroughs.