PROSE: ENHANCING SAMPLE EFFICIENCY IN CODE GENERATION VIA PLAN-AWARE ROLLOUT REORGANIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Group Relative Policy Optimization (GRPO) and related execution-based RL methods for code generation subtract a flat group-relative baseline from every rollout in a group; this baseline is variance-suboptimal as soon as rewards carry latent-plan structure. Programs instantiating the same latent plan tend to succeed or fail together, so execution rewards follow a one-way random-effects mixture whose within-plan intraclass correlation (ICC) ρ_r is ANOVA-measurable. GRPO's i.i.d. rollouts remain marginally exchangeable, so the deficiency lies in the estimator and not in the sampling, but under this mixture the efficient subtractive baseline is plan-conditioned rather than flat. We propose PROSE, which couples explicit plan clustering, posterior-UCB budget reallocation across plans, and an allocation-aware training objective through a single shared plan posterior. Its hierarchical advantage is the uniformly minimum-variance unbiased (UMVU) estimator within the symmetric subtractive class, with a variance-reduction fraction set by ρ_r. We test this prediction directly: at the training operating point the theory predicts an advantage-variance ratio of 0.262 against the flat estimator recomputed on the same batch; the measured value on stored rollouts is 0.26. On LiveCodeBench pass@1, PROSE at $N{=}16$ exceeds GRPO+CoT at $N{=}32$, strict superiority at half the execution budget; PROSE at $N{=}8$ exceeds GRPO+CoT at $N{=}16$; against vanilla GRPO, and in execution budget only, PROSE at $N{=}8$ matches GRPO at $N{=}32$ under a pre-specified equivalence margin of $\delta{=}1.0$ pass@1. On mathematical reasoning, where the ICC is lower, the gain is correspondingly smaller, as the diagnosis predicts.

1 INTRODUCTION

Large language models perform strongly on code generation benchmarks scored by execution (Ni et al., 2024; Chen et al., 2021; Li et al., 2024), and reinforcement learning (RL) post-training (Ouyang et al., 2022) with grouped-sampling objectives such as GRPO (Guo et al., 2025; Gehring et al., 2024) is the standard route to improving them (Le et al., 2022), yet severely sample-inefficient: execution rewards are extremely sparse, so most sampled programs receive zero reward, leaving high-variance advantage estimates and weak optimization signal at any fixed execution budget (Dou et al., 2024; Bamba et al., 2025; Xu et al., 2025; Ridnik et al., 2024).

We locate one structural source of that inefficiency in the estimator rather than in the sampler. Under GRPO's i.i.d. sampling the N rollouts of a group are marginally exchangeable and the unconditional covariance between two draws is zero; nothing about the sampling is violated. What is structured is the reward distribution. Code generation is shaped by latent reasoning plans (Jiang et al., 2024; Lei et al., 2025), and programs instantiating the same plan tend to succeed or fail together, so rewards follow a one-way random-effects mixture with strictly positive within-plan intraclass correlation (ICC) ρ_r. Under that mixture the flat group-relative baseline $\bar{r}$, efficient in the homogeneous case, is variance-suboptimal; the efficient subtractive baseline is the plan-conditioned one (Theorem 4.1(b)). One-way ANOVA measures ρ_r directly: $\rho_r \approx 0.72$ on APPS+ and $\rho_r \approx 0.48$ on GSM8K (Appendix K). Section 3.2 makes this precise and reports ρ_r in sample-size units via a Kish design effect (Kish, 1965). That is a counterfactual clustered-sampling view, not a claim that GRPO cluster-samples. The same structure recurs at the gradient level, where rollouts of one plan also share score directions

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

and GRPO's gradient variance is inflated by a gradient design effect (Theorem 4.1(c)); we treat that route as design rationale, since its correlation admits only a proxy. The practical cost is wasted execution: when a plan is a dead end its programs fail together, and a uniform allocation keeps paying for executions whose outcomes are already predictable from their siblings.

Three components are therefore needed together: (i) making latent plans explicit so the correlation structure is visible and estimable; (ii) redistributing the execution budget across plans rather than uniformly within a group, so rollouts are no longer spent on redundant dead ends; and (iii) adapting the training objective to the induced non-uniform allocation, so that gradient signal is correctly scaled under the sampling distribution the allocation creates. No one of them suffices alone: explicit plans without reallocation leave the budget where it was, and reallocation without a corrected objective biases the gradient toward over-sampled plans. PROSE realizes all three through a single signal, a per-plan Beta–Bernoulli posterior whose mean and uncertainty jointly govern forward budget allocation and backward credit assignment. Because both passes read the same posterior, they are coupled by construction: the non-uniform allocation chosen in the forward pass is automatically reflected in the gradient scaling of the backward pass. Although instantiated for code generation, the inefficiency PROSE targets, latent-strategy reward correlation within a group, arises in any grouped-sampling RL setting whose rollouts share latent strategies, such as mathematical reasoning and tool-use agents (Appendix Q).

- **A measurable diagnosis with a matching optimality result.** PROSE's hierarchical advantage is the uniformly minimum-variance unbiased (UMVU) estimator within the symmetric subtractive class, with variance-reduction fraction set by the measured ρ_r (Theorem 4.1). The qualifier matters: process-level and partial-credit rewards leave this class, so the claim does not extend to them.
- **A quantitative prediction for the core mechanism.** At the training operating point ($k{=}M{=}4$, $N{=}16$), Theorem 4.1(a) predicts an advantage-variance ratio of 0.262 against the flat advantage on the same clustered batch; recomputing both estimators on the stored rollouts gives 0.26.
- **Empirical gains with an explicit attribution ladder.** On LiveCodeBench pass@1 with Qwen2.5-Coder-Instruct-7B, PROSE at $N{=}16$ exceeds GRPO+CoT at $N{=}32$: strict superiority at half the execution budget; PROSE at $N{=}8$ exceeds GRPO+CoT at $N{=}16$. Against vanilla GRPO, and in execution budget only, PROSE at $N{=}8$ is equivalent to GRPO at $N{=}32$ under a two one-sided tests (TOST) procedure with a pre-specified margin $\delta{=}1.0$ pass@1.[1] We decompose the $+5.3$ gain over GRPO into $+1.3$ from explicit plan generation, $+1.9$ from the hierarchical advantage (the component the theory covers), and $+2.1$ from posterior-UCB allocation with ω-weighting and warm-up.

2 RELATED WORK

RL with execution feedback. Correctness in code generation is verifiable by unit tests, making execution a prominent training signal. AlphaCode (Li et al., 2022), CodeRL (Le et al., 2022) and RLTF (Liu et al., 2023) optimize directly against it, while StepCoder (Dou et al., 2024) and RLEF (Gehring et al., 2024) mitigate reward sparsity through curricula or richer feedback. Unlike auxiliary test-case synthesis (Jiang et al., 2025; Wang et al., 2025) or knowledge adaptation (Wu et al., 2025), PROSE leaves the reward unchanged and restructures the group over which the baseline is computed.

Process-level supervision and credit assignment. PRLCoder (Ye et al., 2025) uses fine-grained process supervision, Posterior-GRPO (Fan et al., 2025) posterior signals for token-level credit, and ARIA (Yang et al., 2025) intention-level reward aggregation. These methods change what is rewarded inside a rollout. PROSE's posterior is instead over plans and sets how many rollouts each plan receives, acting on the group's composition rather than on credit within a fixed group.

Planning and structured reasoning. PLANSEARCH (Lei et al., 2025), self-planning (Jiang et al., 2024), planning-before-coding (Zhang et al., 2025), tree-structured search (Yao et al., 2023) and

[1] The $\delta{=}1.0$ margin was fixed a priori as the minimal practically relevant pass@1 difference, before the $N{=}8$ versus $N{=}32$ comparison was examined; it was not registered on an external platform.

multi-agent systems (Islam et al., 2024) all make plans explicit, but improve individual generations at inference time rather than training-time rollout organization. In our attribution ladder, plan prompting alone (GRPO+CoT) accounts for $+1.3$ of PROSE's $+5.3$ gain.

Rollout selection and reweighting. REINFORCE-ADA (Xiong et al., 2025), XRPO (Bamba et al., 2025) and Xu et al. (2025); Zheng et al. (2025); Yao et al. (2025) reweight or prune individual rollouts, conditioning on observable per-rollout statistics (difficulty, reward density, redundancy) without modelling the latent variable that makes rollouts correlated. PROSE operates at a coarser granularity: it makes the latent plan explicit, estimates plan values per cluster, and reallocates budget across rather than within plans. This addresses the latent-plan reward correlation itself rather than its per-rollout symptoms. Table 8 compares these methods on three design axes and one cost axis; Appendix Q gives the full per-method comparison.

3 PROBLEM FORMULATION

3.1 LATENT PLAN VIEW OF CODE GENERATION

A policy π_θ samples a program $y \sim \pi_\theta(\cdot \mid x)$ for an input x, executed against tests for a sparse binary reward $r(y) \in \{0, 1\}$. We posit a latent reasoning plan $p \in \mathcal{P}$ (Wei et al., 2022), so that generation factorizes as $p \sim \pi_\theta(p \mid x)$, $y \sim \pi_\theta(y \mid x, p)$, with p externalized as a short textual plan by a planning prompt (Appendix S) and k plans drawn per input. Rollout success is then largely determined by the plan rather than by token-level sampling noise, which makes the plan the right granularity for credit assignment (Appendix D.1).

3.2 LATENT-PLAN REWARD CORRELATION AND FLAT-BASELINE SUBOPTIMALITY

GRPO baselines the policy gradient on the group mean of N i.i.d. rollouts from the plan-marginal $\pi_\theta(y \mid x)$ (Guo et al., 2025). Nothing about this sampling is defective: the N draws are marginally exchangeable, their unconditional covariance is exactly zero, and the group mean satisfies $\mathrm{Var}(\bar{r}) = \sigma^2/N$. What the latent plan changes is the distributional structure of that marginal, and hence which baseline is efficient. Writing the induced reward law as a one-way random-effects mixture over plans,

$$r_{i,j} = v_i + \varepsilon_{i,j}, \qquad v_i := \mathbb{E}[r \mid x, p_i], \qquad \mathrm{Var}(v_i) = \sigma_B^2, \qquad \mathrm{Var}(\varepsilon_{i,j}) = \sigma_W^2, \tag{1}$$

with $\sigma^2 = \sigma_B^2 + \sigma_W^2$ and intraclass correlation (ICC) $\rho_r := \sigma_B^2/\sigma^2 \in (0, 1)$. Rollouts sharing a plan have covariance $\sigma_B^2 > 0$; independent rollouts have none. Both hold under i.i.d. sampling, and it is the former, a property of the reward distribution rather than of the sampler, that a group-relative baseline must contend with.

The consequence is at the advantage level. GRPO's flat advantage $\widehat{A}_j = r_j - \bar{r}$ pools its baseline across plans, leaving σ_B^2 (the part of the return that the realized plan already predicts) inside the advantage. It is the efficient subtractive control variate precisely in the homogeneous case $\rho_r = 0$; once $\rho_r > 0$ it is variance-suboptimal, and the efficient choice is the plan-conditioned baseline $\widehat{V}_i$ of Equation (1). Theorem 4.1(a)–(b) makes this exact: the code-level advantage has variance $\frac{M-1}{M}(1-\rho_r)\sigma^2$ and is minimum-variance unbiased *within the symmetric subtractive class*; Section 5 tests its protocol-matched variance reduction on stored rollouts. The defect is a mis-specified control variate rather than a broken sampler, and reorganizing rollouts at training time repairs it.

Design-effect diagnostic. To read ρ_r in sample-size units, consider the counterfactual design in which the same rollouts were drawn as k plans of $M{=}N/k$ implementations; for such a clustered mean the Kish design effect (Kish, 1965)

$$D_r = 1 + (M{-}1)\rho_r, \qquad M := N/k, \tag{2}$$

gives effective sample size $N_{\text{eff}} = N/D_r$ against a nominal N. Equation (2) is not a claim that GRPO performs two-stage cluster sampling, which it does not, nor a variance-inflation claim about its i.i.d. group mean; it reports, in sample-size units, what fraction of a group's reward variance is plan-level. The counterfactual is the operative one because clustered organization is what PROSE adopts (Appendix D.3). At $\rho_r{\approx}0.72$, $M{=}4$: $D_r \approx 3.16$.

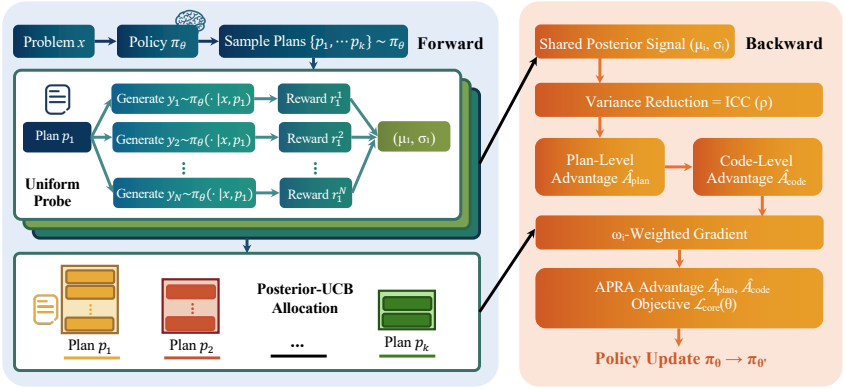

Figure 1: Overview of PROSE: plans are probed uniformly; a per-plan Beta–Bernoulli posterior then drives forward budget reallocation (posterior-UCB) and backward gradient weighting.

The same structure acts again at the gradient level, through $D_{\text{eff}} = 1 + (M-1)\rho_h^{\text{flat}}\rho_r$ (Theorem 4.1(c)); we read that route as design rationale, since ρ_h^{flat} admits only a cosine proxy (Appendix D.4). Empirically $\rho_r \approx 0.72$ on APPS+ and 0.48 on GSM8K (Appendix K): the presumed structure is measurable in both domains, in the severity order the analysis predicts.

4 PLAN-AWARE ROLLOUT REORGANIZATION

PROSE uses plan posterior statistics to govern forward budget allocation and backward credit assignment through a single shared signal (Figure 1). Full pseudocode is in Appendix C (Algorithms 1 and 2); among related GRPO variants only PROSE couples plan structure, adaptive budget and an allocation-aware objective (Table 8).

4.1 POSTERIOR-COUPLED ALLOCATION AND CREDIT ASSIGNMENT

Given an input x, we sample k plans $\{p_i\}_{i=1}^k$ under a fixed total execution budget N; plan i produces implementations $\{y_{i,j}\}_{j=1}^{M_i}$ with rewards $r_{i,j}$ and $\sum_i M_i = N$. The balanced comparator $M := N/k$, the realized count M_i and the Stage-1 probe $M^{(0)}$ are distinguished in Appendices A and E.1.

Plan posterior from a uniform probe. We first spend $M^{(0)}$ executions on every plan; with Bernoulli rewards the conjugate update (Russo et al., 2018) gives plan i the posterior $\text{Beta}(\alpha_0+s_i,\ \beta_0+M^{(0)}-s_i)$ from its s_i successes. Its mean μ_i and standard deviation σ_i (Appendix A) are the only per-plan statistics the method carries.

Posterior-coupled forward allocation. The remaining budget follows a UCB-style posterior score $u_i = \mu_i + \lambda\sigma_i$ (Auer et al., 2002) through softmax-mixture probabilities

$$q_i = (1-\eta)\cdot \frac{\exp(u_i/\tau_{\text{alloc}})}{\sum_{\ell=1}^k \exp(u_\ell/\tau_{\text{alloc}})} + \frac{\eta}{k}, \tag{3}$$

with largest-remainder rounding to counts $M_i^{(1)}$ summing to $N_{\text{rem}} = N - kM^{(0)}$, and $M_i = M^{(0)} + M_i^{(1)}$. The floor η bounds every allocation probability away from zero, which keeps the bias bound of Theorem F.1 finite.

Allocation-coupled backward update. Per plan we form $\widehat{V}_i = M_i^{-1}\sum_j r_{i,j}$ and $\bar{V} = k^{-1}\sum_\ell \widehat{V}_\ell$, giving the hierarchical advantages $\widehat{A}_{\text{plan},i} = \widehat{V}_i - \bar{V}$ and $\widehat{A}_{\text{code},i,j} = r_{i,j} - \widehat{V}_i$. The realized allocation ratio $\omega_i = M_i/(N/k)$ aligns backward credit with forward budget commitment; it is deliberately not the importance correction $1/\omega_i$, which would undo that signal (Appendix E.2). With the standard GRPO clipped surrogate $\phi(\hat{r}, A)$ (Guo et al., 2025),

$$\mathcal{L}_{\text{core}}(\theta) = \mathbb{E}_{\mathcal{D}}\Big[\sum_i \omega_i\, \phi\big(\hat{r}_i^{\text{plan}}, \widehat{A}_{\text{plan},i}\big) + \sum_i\sum_j \phi\big(\hat{r}_{i,j}^{\text{code}}, \widehat{A}_{\text{code},i,j}\big) - \beta\sum_i \text{KL}_i\Big], \tag{4}$$

where $\hat{r}_i^{\text{plan}}$ and $\hat{r}_{i,j}^{\text{code}}$ are the usual policy ratios at the plan and code levels and KL_i the code-level reference KL (written out in Appendix E.3). At $\omega_i \equiv 1$ this recovers the uniform-allocation objective.

4.2 Theoretical Guarantees

We compare PROSE's hierarchical advantage against GRPO's flat baseline under balanced allocation ($M_i = M = N/k, \omega_i = 1$), exactly the PVE stage (Table 1); the APRA bias bound (Theorem F.1), the gradient derivation (Appendix H) and the iteration-complexity analysis (Appendix J) are deferred.

Theorem 4.1 (Advantage-Variance Reduction and Optimality of the Hierarchical Advantage)**.** *Adopt the one-way random-effects model $r_{i,j} = v_i + \varepsilon_{i,j}$ with ICC $\rho_r := \sigma_B^2/\sigma^2 \in (0,1)$, balanced plan size $M = N/k$, and $\sigma^2 = \sigma_B^2 + \sigma_W^2$.*

(a) Advantage-variance reduction. *A flat grand-mean advantage computed on a batch organized as k plans of $M = N/k$ rollouts each has variance $\text{Var}(\widehat{A}_{i,j}^{\text{flat}}) = \frac{k-1}{k}\sigma_B^2 + \frac{N-1}{N}\sigma_W^2$ (one-way ANOVA identity; Appendix G), while* PROSE*'s code-level advantage $\widehat{A}_{\text{code},i,j} = r_{i,j} - \widehat{V}_i$ has variance $\frac{M-1}{M}\sigma_W^2 = \frac{M-1}{M}(1-\rho_r)\sigma^2$. The identity admits two instantiations of the flat comparator, labeled throughout:*

$$\begin{aligned} \text{same-batch:}\ R_{\text{same}} &= \frac{\frac{M-1}{M}(1-\rho_r)}{\frac{k-1}{k}\rho_r + \frac{N-1}{N}(1-\rho_r)} = 0.262, \\ \text{as-deployed:}\ R_{\text{dep}} &= \frac{(M-1)(1-\rho_r)\,N}{M(N-1)} = 0.224, \end{aligned} \tag{5}$$

both at the operating point $k{=}M{=}4$, $N{=}16$, $\rho_r{=}0.72$ (Appendix K). R_{same} is the ratio estimated when both estimators are recomputed on one stored clustered batch of k plans of size M; R_{dep} divides the same $M{=}4$ numerator by the i.i.d. deployment comparator ($k{=}N$, $M{=}1$), where the identity reduces exactly to $\frac{N-1}{N}\sigma^2$. The variance-reduction fractions $1-R$ are 0.738 and 0.776.

(b) Optimality within the symmetric subtractive class. *Among symmetric subtractive advantages $\widehat{A}_{i,j} = r_{i,j} - b(\{r_{i,m}\}_{m=1}^{M})$ with b permutation-invariant within a plan and plan-conditionally unbiased for v_i, the code-level advantage is the uniformly minimum-variance unbiased (UMVU) estimator* within the symmetric subtractive class*, with second moment $(M{-}1)\sigma_W^2/M$; the argument applies Lehmann–Scheffé directly to the complete sufficient statistic $\widehat{V}_i$ (Theorem I.1).*

(c) Gradient-level extension *(Appendix H). Under **(A1)** cross-plan score orthogonality, **(A2)** $\rho_h^{\text{flat}} \geq 0$ and **(A3)** $\|h^{\text{cond}}\|^2 \geq (M{-}1)(1-\rho_h^{\text{flat}})\|h^{\text{plan}}\|^2$ (sufficient; the exact requirement carries an extra factor $1 + (M{-}1)\rho_h^{\text{flat}}$ on the left), the per-step policy-improvement ratio satisfies $\Delta J^{\text{PROSE}}/\Delta J^{\text{GRPO}} \geq 1$, with GRPO's gradient variance inflated by the gradient design effect $D_{\text{eff}} = 1 + (M{-}1)\rho_h^{\text{flat}}\rho_r$ against the counterfactually clustered comparator ($D_{\text{eff}} \to 1$ for the as-deployed one). The same assumptions carry the iteration-complexity corollary (Theorem J.1).*

Remark 4.2 (Fixed-M asymptotics). For both instantiations, with M fixed and the number of plans growing, $1 - R \to [1 + (M{-}1)\rho_r]/M \geq \rho_r$, equality only at $\rho_r = 1$; reaching ρ_r requires $M \to \infty$ as well. At our operating point ($M{=}4$, $\rho_r{\approx}0.72$) the fixed-M limit is 0.79, so the finite-M regime overstates the asymptotic ρ_r rather than understating it.

The direct quantitative test. Part (a) removes the between-plan component σ_B^2, leaving only the within-plan residual. Because our measurement recomputes both estimators on one stored clustered batch at the operating point ($k{=}M{=}4$, $N{=}16$), the protocol-matched prediction is the same-batch instantiation: $R_{\text{same}} = 0.262$ against a measured 0.26, with no free parameter, while $R_{\text{dep}} = 0.224$ answers a different, cross-design question (Section 5). Since ρ_r and the measured ratio come from the same rollouts, this agreement is in-sample in its parameter (Appendix K.4). This variance ratio, not a pass@1 increment, is the theory's direct quantitative test; the gradient design effect translates into accuracy only as a secondary sanity check (Section 5). Part (b) certifies that no other plan-membership baseline in this class improves further. Its qualifier matters: process-level or partial-credit rewards (Ye et al., 2025) leave the symmetric subtractive class, so the optimality claim does not extend to them.

Empirical status of (A3). (A3) is the one gradient-level condition checkable at current model scales. At $\widehat{\rho}_h^{\text{flat}} = 0.56$ and $M = 4$ the sufficient form requires a score-norm ratio $\|h^{\text{cond}}\|^2/\|h^{\text{plan}}\|^2 \geq 1.32$, the exact form only ≥ 0.49. Two proxies for that ratio disagree: a token-count reading of the

Table 1: Theory-to-method map: what is proved at each stage, what the proof assumes, and what is only empirical. Reduction fractions are given for both labeled comparators of Theorem 4.1(a) at $k{=}M{=}4$, $N{=}16$, $\rho_r{=}0.72$.

Stage	What is proven	Key assumptions	What is empirical
PVE ($\omega_i{=}1$)	reduction fraction 0.776 (as-deployed) / 0.738 (same-batch); UMVU *within the symmetric subtractive class* (Theorem 4.1a,b)	one-way random effects; Bernoulli rewards; balanced M	none beyond model fit
APRA ($\omega_i{\neq}1$)	bias at most quadratic in plan-value spread; improvement stays positive after warm-up (Theorem F.1)	calibrated posterior; high-temperature regime $c < 1$	magnitude of the net benefit ($+2.1$ pass@1)
Gradient extension	$\Delta J^{\mathrm{PROSE}}/\Delta J^{\mathrm{GRPO}} \geq 1$; $D_{\mathrm{eff}} = 1 + (M{-}1)\rho_h^{\mathrm{flat}}\rho_r$ (Theorem 4.1c)	(A1) cross-plan orthogonality (approx.); (A2) $\rho_h^{\mathrm{flat}} \geq 0$; (A3) score-norm condition (proxy-checked; Appendix H.7)	$\widehat{\rho}_h^{\mathrm{flat}} = 0.56$ (hidden-state cosine proxy)
Convergence corollary	relative iteration-complexity speed-up (Theorem J.1)	L-smooth + μ-PL; (A1)–(A3)	—

norm heuristic ($\sim$35 plan versus $\sim$200+ code tokens) gives $\approx$5.7, whereas our own ρ_h estimates imply $\approx$1.3 (Appendix H.7). The exact form is cleared under both readings, the sufficient form only under the first, so we carry (A3) in its exact form: proxy-checked rather than assumed.

Scope and limitations. Theorem 4.1 covers the PVE stage directly (uniform allocation, unbiased estimators); under APRA, non-uniform allocation introduces bias bounded by Theorem F.1, and the per-step improvement degrades but stays positive once warm-up has calibrated the posterior. Part (c) rests on assumptions of unequal strength: (A1) is approximate, since rollouts share the problem-description prefix (Appendix H derives the cross-plan term, $\widehat{\rho}_{\mathrm{cross}} \approx 0.08$, and finds it conservative), while (A2)–(A3) are checkable only through the cosine proxy. The theory thus supplies a design rationale (why hierarchical advantages dominate flat baselines, and which measurable quantity, ρ_r in Section 3.2, says when) plus one falsifiable prediction, the variance ratio; it does not predict improvement magnitude, and the iteration-complexity result (Theorem J.1) compares PL-derived sufficient-iteration bounds rather than absolute complexities.

4.3 Two-Stage Training Schedule and Implementation

PROSE trains in two stages. **Plan-value estimation (PVE) warm-up** uses uniform plan sampling ($M_i{=}M$, hence $\omega_i{=}1$), the regime Theorem 4.1 covers exactly, and calibrates the posterior before committing to noisy plans. **Adaptive plan-conditioned rollout allocation (APRA)** then activates posterior-coupled allocation (Equation (3)) and ω_i-weighted gradients in Equation (4). The switch occurs at step 500 of a $1{,}000$-step run (Appendix R); two optional extensions are in Appendix B.

Implementation cost. PROSE modifies an existing GRPO trainer rather than replacing it: $\approx$200 lines over a VeRL GRPO baseline. The minimal recipe (plan prompting plus the hierarchical advantage under uniform allocation, i.e. PVE only) is a $\approx$100-line change that already captures $+3.2$ of the $+5.3$ total LiveCodeBench gain (60%), of which the hierarchical advantage alone contributes $+1.9$ (Section 5). The posterior and UCB updates add no backward passes (Appendix E.6); wall-clock overhead is in Table 4.

Table 2: Main results with **Qwen2.5-Coder-Instruct-7B**; in-house rows are mean ± std over 3 seeds at the matched budget $N=16$. The bottom block reports released external checkpoints under our protocol: *reference points; not budget- or training-matched*.

	HumanEval		MBPP		LiveCodeBench	
Approach	**Pass@1**	**Pass@5**	**Pass@1**	**Pass@5**	**Pass@1**	**Pass@5**
Base	88.4±0.2	94.2±0.2	83.5±0.3	90.9±0.2	16.9±0.4	29.2±0.4
GRPO	88.7±0.2	94.1±0.2	83.7±0.3	91.0±0.2	17.1±0.4	29.5±0.4
GRPO + CoT	88.9±0.3	94.5±0.2	84.2±0.3	91.3±0.2	18.4±0.4	31.0±0.4
PVE	90.2±0.2	95.1±0.2	85.5±0.2	92.0±0.2	20.3±0.3	33.4±0.3
APRA	90.5±0.3	95.3±0.2	85.8±0.2	92.2±0.2	21.0±0.4	34.2±0.3
PROSE	**91.5±0.2**	**96.2±0.2**	**86.8±0.2**	**93.4±0.2**	**22.4±0.3**	**37.1±0.3**
External checkpoints (Xie et al., 2025; Zeng et al., 2025; Open R1 / Hugging Face, 2025): reference points; not budget- or training-matched						
Dream-Coder	82.9	91.4	79.6	88.7	18.9	31.4
AceCoder	89.0	94.6	85.1	91.9	20.4	33.8
OlympicCoder	78.7	89.2	76.3	86.9	19.1	31.7

Table 3: Attribution of the $+5.3$ LCB pass@1 gain over GRPO at $N=16$; each row adds one mechanism.

Step (LCB pass@1, $N=16$)	Δ	**Mechanism**	**Status w.r.t. theory**
GRPO 17.1 → CoT 18.4	$+1.3$	plan prompting	outside the theory (taken as given)
CoT 18.4 → PVE 20.3	$+1.9$	hierarchical advantage	Theorem 4.1(a)/(b); tested via the variance ratio
PVE 20.3 → PROSE 22.4	$+2.1$	posterior-UCB allocation	Theorem F.1 bias bound; magnitude empirical

5 EVALUATION

5.1 EXPERIMENTAL CONFIGURATION

We post-train instruction-tuned code LLMs, using Qwen2.5-Coder-Instruct-7B (Hui et al., 2024) as default (3B in Appendix O.1), training on APPS+ (Dou et al., 2024) and evaluating on HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021) and LiveCodeBench (LCB) (Jain et al., 2024), the latter restricted to a contamination-free window (1 September–1 December 2024). All methods share the execution budget $N=16$ per input, 1,000 optimization steps from a common base checkpoint, and identical sampling settings: generation temperature is standardized at $\tau_{\text{plan}}=\tau_{\text{code}}=0.7$ and top-$p=1.0$ across all methods, for training and evaluation alike (Appendix R).

5.2 MAIN RESULTS AND ATTRIBUTION

In Table 2, PROSE attains the best budget-matched pass@1 and pass@5 on all three benchmarks. The largest gain is on LCB: $+5.3$ over GRPO at the same $N=16$ ($17.1 \rightarrow 22.4$, $+31.0\%$ relative pass@1 gain). Against GRPO+CoT (same plan scratchpad, uniform allocation, flat baseline), PROSE gains $+2.6$, $+2.6$ and $+4.0$ (paired bootstrap, 10,000 resamples, $p<0.01$ on all three benchmarks), so plan prompting alone does not account for it. Appendix P.1 reads the near-ceiling benchmarks and the pass@5 ordering.

Where the gain comes from. Table 3 decomposes the $+5.3$: $\approx 25\%$ ($+1.3$) is plan prompting; 36% ($+1.9$) is the ICC-diagnosed hierarchical-advantage mechanism; 40% ($+2.1$) is adaptive allocation. That last share bounds what generic adaptive allocation buys: posterior-UCB over random groups, which carry near-zero ICC, yields only $+1.2$ ($\approx 23\%$ of the headline; Appendix P.4), so the rest of the allocation step needs groups that track latent plans—the condition the variance argument also needs.

5.3 DIRECT TEST OF THE VARIANCE-REDUCTION PREDICTION

Theorem 4.1(a) makes a quantitative prediction that does not pass through pass@1 and is directly falsifiable. On a stored training batch organized as $k=4$ plans of $M=4$ rollouts ($N=16$, the operating point), the ANOVA identity at $\rho_r=0.72$ predicts a plan-conditioned-to-flat advantage-variance ratio

Table 4: Compute accounting in three units we never conflate; wall-clock is for 1,000 steps on identical hardware. "—": not logged at that budget. †Implied by the two measured overheads, not separately measured.

Method	Exec. budget N	Gen. tokens / problem	Wall-clock (rel. GRPO)	LCB pass@1
GRPO	16	6.1k	1.000× (ref.)	17.1±0.4
GRPO + CoT	16	6.5k	1.031×†	18.4±0.4
GRPO + CoT	32	—	—	21.1±0.4
PROSE	16	**6.3k**	1.050×	**22.4±0.3**

R_{same}=0.262, i.e. a 73.8% variance reduction; recomputing both estimators on the same stored rollouts gives **0.26**, matching to the precision of the ICC estimate. The comparator is the *same-batch* flat advantage; the *GRPO-as-deployed* instantiation (k=N, M=1) answers a different, cross-design question and gives R_{dep}=0.224 (Appendix G.3). Since ρ_r and the measured ratio come from the same rollouts, the agreement is in-sample in its parameter; a split-half version (ρ_r fitted on one half of the problems, the prediction evaluated on the disjoint half) is specified as a safeguard and not yet run (Appendix K).

As a secondary sanity check (not a theorem-level prediction of pass@1), translating the asymptotic D_{eff}=1 + $(M-1)\hat{\rho}_h^{\text{flat}}\rho_r \approx 2.21$ (finite batch 1.68, Remark H.4) through the measured GRPO+CoT budget curve of Table 25 gives $\approx$ +2.2 pass@1 against the +1.9 observed for the hierarchical-advantage step, while the reward-level route (D_r=3.16) over-predicts at +3.2, as Theorem 4.1(c) requires of the ρ_h^{flat}=1 upper bound. The ordering of the two routes is the check, and no claim depends on either value (Appendix P.3).

Post-hoc credit assignment versus forward plan conditioning. A competing explanation for the +1.9 CoT→PVE step is that plan conditioning simply makes generation easier. **PVE-posthoc** separates that from the baseline-efficiency account: the rollouts are exactly GRPO+CoT's and only credit assignment changes, on plans extracted post hoc from the generated code and clustered (Appendix M). It reaches **19.7** LCB pass@1 against 18.4 for GRPO+CoT and 20.3 for forward PVE: retrospective hierarchical credit assignment on i.i.d.-generated rollouts recovers +1.3 of the +1.9, and the residual 0.6 measures the added value of coupling generation to shared plans. The mechanism is thus principally a correction to latent-plan reward correlation already present in ordinary GRPO+CoT rollouts, not an artifact of an easier conditioned task.

5.4 Budget Scaling, Re-scoped Efficiency Claims, and Compute Accounting

The claim that survives. Across $N \in \{8, 16, 32\}$, PROSE dominates both baselines at every budget and the gap widens in N (full ladder and curve: Table 25 and Figure 3). At N=16, PROSE (22.4±0.3) strictly exceeds GRPO+CoT run at twice that budget, N=32 (21.1±0.4): superiority at half the execution budget, with no equivalence margin involved (Welch $t \approx 4.5$, 3 seeds). The pattern repeats one budget lower, where PROSE at N=8 (20.1±0.3) exceeds GRPO+CoT at N=16 (18.4±0.4), $t \approx 5.9$. Where both curves are measured PROSE sits ≈3.0× left of GRPO+CoT; larger figures extrapolate past N=32 (Appendix P.2).

What is withdrawn. Against vanilla GRPO, PROSE@N=8 (20.1) versus GRPO@N=32 (19.8) differs by +0.3 (p=0.41, two-sided), and a two one-sided tests (TOST) equivalence procedure with the *pre-specified* margin δ=1.0 pass@1 places the difference inside the equivalence bounds (p_{TOST}<0.05). That 4× statement is scoped to execution budget and to vanilla GRPO, and is made nowhere else in the paper. It does not hold against GRPO+CoT: PROSE@N=8 (20.1) is 1.0 pass@1 *below* GRPO+CoT@N=32 (21.1), so we make no 4× claim against the stronger baseline. The replacement is margin-free strict superiority at half the budget against the stronger baseline, rather than margin-dependent equivalence at a quarter of the budget against the weaker one. At N=8, PROSE uses k=4, $M^{(0)}$=1, leaving four post-probe executions, so that row is not a degenerate PVE-only run.

Compute accounting. Table 4 keeps execution budget, token generation and wall-clock apart. At the matched $N{=}16$, PROSE generates fewer tokens per problem (6.3k) than GRPO+CoT (6.5k) despite an explicit planning stage, because $k{=}4$ plans are amortized over 16 rollouts whereas GRPO+CoT re-derives one inside every rollout; matched-success audits (5.1% fewer tokens at equal success counts) are in Appendix O.9. Wall-clock overhead over $1{,}000$ steps is $+5.0\%$ over vanilla GRPO and $+1.8\%$ over GRPO+CoT.

5.5 Ablations and Generalization

Plan structure versus allocation. In a separate progressive ablation, semantic plans under uniform allocation capture $+3.5$ of the $+5.3$ (66%; Appendix P.4) and posterior-UCB over random groups adds $+1.2$. The combined $+5.3$ exceeds the sum of the marginals by $+0.6$, but that contrast has standard error 0.44 ($t \approx 1.38$) at 3 seeds and is not resolvable: we claim only that the mechanisms compose at least additively. The design is not a complete 2×2; the flat cell is plain GRPO, not a Random | Uniform control, so the $+1.2$ marginal confounds allocation rule with grouping structure.

Cross-domain and cross-family. Latent-plan reward correlation arises wherever rollouts share a latent strategy. On GSM8K (Cobbe et al., 2021), the ICC is $\rho_{\mathrm{math}} \approx 0.48 \pm 0.16$ ($D_r \approx 2.44$), below code (0.72 ± 0.14), and PROSE gains $+1.6$ over GRPO and $+1.2$ over GRPO+CoT—smaller, as Eq. (2) predicts, and further compressed by the $92\%+$ ceiling; an ordering test only, ρ_h^{flat} having not been measured there. On DeepSeekCoder-7B-Instruct (Guo et al., 2024), PROSE yields $+3.7$ LCB over GRPO+CoT against $+4.0$ on Qwen, with $\rho_r \approx 0.69$, so Eq. (2) transfers without retuning (Appendices P.5 and P.6).

Where the remaining analyses live. Gains rise with difficulty and with the planning content of the task ($+5.0$ on dynamic programming to $+2.2$ on implementation; Appendix O.7) and track plan diversity, rising monotonically across within-problem diversity terciles ($+2.1/+3.8/+5.2$ over CoT) and falling to $+2.4$ where a near-duplicate pair is present (Appendix O.6). Extending APRA is monotone ($21.3/21.9/22.4$ at steps $600/800/1000$; Appendix O.8), not an artifact of the stopping point. Baseline ICCs use plans recovered retrospectively from generated code by an auxiliary LLM (unconditioned GRPO emits no plan text) and stay in 0.36–0.44 across variants (Appendix N). Execution-noise robustness (where PROSE stays ahead of the baselines at every noise level), allocation-rule ablations, hyperparameter sensitivity and the training-reward diagnostic are in Appendices O and P.

6 Conclusion

We introduced PROSE, a plan-aware rollout reorganization framework resting on a measurable distributional diagnosis rather than a sampling defect: execution rewards in code RL follow a latent-plan mixture, under which GRPO's flat group-relative baseline is variance-suboptimal (Theorem 4.1(b)). The reward-level ICC ρ_r, measurable by one-way ANOVA ($\rho_r \approx 0.72$ on APPS+), both quantifies the loss and predicts the remedy: the hierarchical advantage is UMVU *within the symmetric subtractive class*, and its predicted same-batch advantage-variance ratio of 0.262 matches the 0.26 recomputed on stored rollouts. PROSE improves LiveCodeBench pass@1 by $+5.3$ over GRPO at $N{=}16$, where its 22.4 ± 0.3 also exceeds GRPO+CoT at twice the execution budget ($N{=}32$, 21.1 ± 0.4); against vanilla GRPO alone, PROSE@$N{=}8$ is TOST-equivalent to GRPO@$N{=}32$ in execution budget under the pre-specified $\delta{=}1.0$ margin.

Limitations. Four scope conditions bound these results. *(i) Model scale.* We evaluate only 3B and 7B policies from two families; whether plan-level reward correlation persists at 70B+, where one rollout may already blend several strategies and compress ρ_r, is untested. *(ii) Task granularity.* All benchmarks are self-contained algorithmic problems with unit tests; repository-level and agentic settings are not implied, since "plan" and reward must be redefined there before ρ_r is well posed (Appendix Q.1). *(iii) Plan diversity.* PROSE degrades toward flat GRPO as plans collapse, and APRA does erode diversity (near-duplicates rise from 9% to 13% over $1{,}000$ steps); the exploration term η floors this loss (Appendix O.6). *(iv) Overhead and resolution.* Plan generation costs $+5.0\%$ wall-clock over vanilla GRPO (Table 4); 3 seeds suffice for the headline contrasts but not for the $+0.6$ factorial interaction, so we will extend headline rows to 5 seeds and release per-seed outputs.

AI USE STATEMENT

In this work, we used generative AI tools for two purposes only: to aid and polish writing, and for retrieval and discovery of related work.

To aid or polish writing. We used LLM-based assistants to improve the readability of text written by the authors: rephrasing sentences, tightening the presentation of the theoretical statements, and copy-editing grammar and style. No passage was accepted as generated; every AI-assisted sentence was checked by the authors against the underlying results.

For retrieval and discovery. We used LLM-based search tools to help identify potentially related work. Every suggested paper was read by the authors, and each citation and its characterization in Section 2 and Appendix Q was verified against the original source.

Tasks for which generative AI was not used. We did not use generative AI tools to generate research ideas or hypotheses, formulate the method, develop the theoretical model, state or prove theorems, design or run experiments, implement the method, clean or generate data, or analyze and interpret results. Every reported number comes from an executed run or from arithmetic on executed runs.

We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work trains code-generation models with execution feedback on public benchmarks. It involves no human subjects, no private or personally identifying data, and no crowdsourcing. The training corpus (APPS+ (Dou et al., 2024), derived from the public APPS dataset (Hendrycks et al., 2021)) and the evaluation suites (HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021), LiveCodeBench (Jain et al., 2024), GSM8K (Cobbe et al., 2021)) are publicly available and consist of standard programming and arithmetic-reasoning problems containing no exploit or vulnerability code; all base models are publicly released checkpoints used within their stated license terms and cited at first use. We release no new dataset or pre-trained model as an asset.

Dual use. PROSE is a sample-efficiency multiplier for execution-based code RL rather than a new capability. Its positive impacts follow from that: a lower compute barrier and reduced energy cost for training capable code models, and correspondingly broader access for compute-constrained groups. The negative impacts follow from the same property in the opposite direction: cheaper code RL accelerates the training of models that emit insecure or low-quality code at scale, and may amplify automation-related labor effects in software engineering. Being an efficiency multiplier, the method proportionally amplifies whichever effects are inherited from the underlying code-generation models.

A risk specific to plan-level budget concentration. One risk is added by the method rather than inherited, and we state it plainly. Where uniform sampling spreads its executions evenly, PROSE deliberately concentrates execution budget on the plans its posterior judges most promising. In a benign setting that concentration is exactly what buys the efficiency. In an adversarial one (an attacker running the same loop against a verifier that scores attack success instead of unit-test pass/fail), the same mechanism would raise the hit rate of a malicious plan by identifying, earlier and with fewer executions, which high-level strategy works. The risk attaches to the allocation rule, not to our reward: any verifier able to score a harmful objective would inherit it, and the effect is larger the more the outcome is determined at the level of strategy rather than of implementation detail, which is precisely the regime the measured ICC identifies. We see no mitigation internal to the algorithm and note it as a property that adopters of plan-level allocation should weigh.

Research integrity. Claims in this paper are scoped to the comparison that supports them. In particular, the budget-equivalence result is reported only against vanilla GRPO and only in execution budget, and Section 5 states explicitly the stronger-baseline comparison in which it does not hold. The $\delta{=}1.0$ TOST margin was pre-specified but was not registered on an external platform, and we describe it as such throughout.

Reproducibility statement

The full assumption sets and complete proofs for every theoretical claim—the ANOVA variance identity underlying Theorem 4.1(a), the minimum-variance optimality statement together with the class restriction it carries, the gradient-level extension with its assumptions (A1)–(A3), the allocation-induced bias bound, and the iteration-complexity corollary—are given in Appendices G to J. The measurement protocols for the two quantities the theory depends on are given separately: the one-way random-effects ICC estimation procedure, sample sizes, and the retrospective plan-extraction pipeline used for unconditioned baselines in Appendix K, and the hidden-state cosine proxy for the within-plan score correlation, with its explicit caveats, in Appendix L. Pseudocode for both training stages appears as Algorithms 1 and 2 in Appendix C, with notation collected in Appendix A; complete per-method hyperparameters, decoding settings, hardware, and the statistical reporting protocol are in Appendix R (Tables 29 and 31); the verbatim plan-generation and code-generation prompt templates are in Appendix S; and extended results, ablations, and robustness checks are in Appendix O. Code, training and evaluation configurations, and per-seed outputs for all in-house runs will be released under a permissive open-source license; they are withheld at submission time only to preserve double-blind anonymity.

References

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002. doi: 10.1023/A:1013689704352.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Udbhav Bamba, Minghao Fang, Yifan Yu, Haizhong Zheng, and Fan Lai. XRPO: Pushing the limits of GRPO with targeted exploration and exploitation. *arXiv preprint arXiv:2510.06672*, 2025.

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

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

Shihan Dou, Yan Liu, Haoxiang Jia, Limao Xiong, Enyu Zhou, Wei Shen, Junjie Shan, Caishuang Huang, Xiao Wang, Xiaoran Fan, et al. StepCoder: Improve code generation with reinforcement learning from compiler feedback. *arXiv preprint arXiv:2402.01391*, 2024.

Lishui Fan, Yu Zhang, Mouxiang Chen, and Zhongxin Liu. Posterior-GRPO: Rewarding reasoning processes in code generation. *arXiv preprint arXiv:2508.05170*, 2025.

Jonas Gehring, Kunhao Zheng, Jade Copet, Vegard Mella, Quentin Carbonneaux, Taco Cohen, and Gabriel Synnaeve. RLEF: Grounding code LLMs in execution feedback with reinforcement learning. *arXiv preprint arXiv:2410.02089*, 2024.

Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5:1471–1530, 2004.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. *arXiv preprint arXiv:2507.17746*, 2025.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y.K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. DeepSeek-Coder: When the large language model meets programming – the rise of code intelligence. *arXiv preprint arXiv:2401.14196*, 2024.

Daya Guo et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with APPS. In *Advances in Neural Information Processing Systems*, 2021.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, Yunlong Feng, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-Coder technical report, 2024. URL https://arxiv.org/abs/2409.12186.

Md Ashraful Islam, Mohammed Eunus Ali, and Md Rizwan Parvez. MapCoder: Multi-agent code generation for competitive problem solving. *arXiv preprint arXiv:2405.11403*, 2024.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination-free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Xue Jiang, Yihong Dong, Lecheng Wang, Zheng Fang, Qiwei Shang, Ge Li, Zhi Jin, and Wenpin Jiao. Self-planning code generation with large language models. *ACM Transactions on Software Engineering and Methodology*, 33(7):1–30, 2024.

Xue Jiang, Yihong Dong, Mengyang Liu, Hongyi Deng, Tian Wang, Yongding Tao, Rongyu Cao, Binhua Li, Zhi Jin, Wenpin Jiao, et al. CodeRL+: Improving code generation via reinforcement with execution semantics alignment. *arXiv preprint arXiv:2510.18471*, 2025.

Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the Polyak–Łojasiewicz condition. In *European Conference on Machine Learning (ECML)*, 2016.

Leslie Kish. *Survey Sampling*. John Wiley & Sons, 1965.

Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu Hong Hoi. CodeRL: Mastering code generation through pretrained models and deep reinforcement learning. *Advances in Neural Information Processing Systems*, 35:21314–21328, 2022.

Erich L Lehmann and Joseph P Romano. *Testing Statistical Hypotheses*. Springer, 3rd edition, 2005.

Chao Lei, Yanchuan Chang, Nir Lipovetzky, and Krista A Ehinger. Planning-driven programming: A large language model programming workflow. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12647–12684, 2025.

Yixuan Li, Julian Parsert, and Elizabeth Polgreen. Guiding enumerative program synthesis with large language models. In *International Conference on Computer Aided Verification*, pp. 280–301. Springer, 2024.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Remi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with AlphaCode. *Science*, 378(6624):1092–1097, 2022. doi: 10.1126/science.abq1158.

Chaoyue Liu, Libin Zhu, and Mikhail Belkin. Loss landscapes and optimization in over-parameterized non-linear systems and neural networks. *Applied and Computational Harmonic Analysis*, 59: 85–116, 2022.

Jiate Liu, Yiqin Zhu, Kaiwen Xiao, Qiang Fu, Xiao Han, Yang Wei, and Deheng Ye. RLTF: Reinforcement learning from unit test feedback. *Transactions on Machine Learning Research*, 2023. URL https://openreview.net/forum?id=hjYmsV6nXZ.

Ansong Ni, Pengcheng Yin, Yilun Zhao, Martin Riddell, Troy Feng, Rui Shen, Stephen Yin, Ye Liu, Semih Yavuz, Caiming Xiong, et al. L2CEval: Evaluating language-to-code generation capabilities of large language models. *Transactions of the Association for Computational Linguistics*, 12: 1311–1329, 2024.

Open R1 / Hugging Face. OlympicCoder-7B: A 7B competitive coding model (Open R1). `https://huggingface.co/open-r1/OlympicCoder-7B`, 2025. Accessed Jan. 10, 2026.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Tal Ridnik, Dedy Kredo, and Itamar Friedman. Code generation with AlphaCodium: From prompt engineering to flow engineering. *arXiv preprint arXiv:2401.08500*, 2024.

Daniel J. Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, and Zheng Wen. A tutorial on Thompson sampling. *Foundations and Trends in Machine Learning*, 11(1):1–96, 2018. doi: 10.1561/2200000070.

Henry Scheffé. *The Analysis of Variance*. John Wiley & Sons, 1959.

Shayle R Searle, George Casella, and Charles E McCulloch. *Variance Components*. John Wiley & Sons, 2006.

Suryansh Singh Sijwali and Suman Saha. SecureCodeRL: Security-aware reinforcement learning for code generation with partial-credit rewards. *arXiv preprint arXiv:2601.01184*, 2026.

Yinjie Wang, Ling Yang, Ye Tian, Ke Shen, and Mengdi Wang. Co-evolving LLM coder and unit tester via reinforcement learning, 2025. URL `https://arxiv.org/abs/2506.03136`.

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

Haoze Wu, Yunzhi Yao, Wenhao Yu, and Ningyu Zhang. ReCode: Updating code API knowledge with reinforcement learning, 2025. URL `https://arxiv.org/abs/2506.20495`.

Zhihui Xie, Jiacheng Ye, Lin Zheng, Jiahui Gao, Jingwei Dong, Zirui Wu, Xueliang Zhao, Shansan Gong, Xin Jiang, Zhenguo Li, et al. Dream-Coder 7B: An open diffusion language model for code. *arXiv preprint arXiv:2509.01142*, 2025.

Wei Xiong, Chenlu Ye, Baohao Liao, Hanze Dong, Xinxing Xu, Christof Monz, Jiang Bian, Nan Jiang, and Tong Zhang. Reinforce-Ada: An adaptive sampling framework for REINFORCE-style LLM training. *arXiv preprint arXiv:2510.04996*, 2025.

Yixuan Even Xu, Yash Savani, Fei Fang, and J Zico Kolter. Not all rollouts are useful: Down-sampling rollouts in LLM reinforcement learning. *arXiv preprint arXiv:2504.13818*, 2025.

Ruihan Yang, Yikai Zhang, Aili Chen, Xintao Wang, Siyu Yuan, Jiangjie Chen, Deqing Yang, and Yanghua Xiao. ARIA: Training language agents with intention-driven reward aggregation. *arXiv preprint arXiv:2506.00539*, 2025.

Jiarui Yao, Yifan Hao, Hanning Zhang, Hanze Dong, Wei Xiong, Nan Jiang, and Tong Zhang. Optimizing Chain-of-Thought reasoners via gradient variance minimization in rejection sampling and RL. *arXiv preprint arXiv:2505.02391*, 2025.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Yufan Ye, Ting Zhang, Wenbin Jiang, and Hua Huang. Process-supervised reinforcement learning for code generation. *arXiv preprint arXiv:2502.01715*, 2025.

Huaye Zeng, Dongfu Jiang, Haozhe Wang, Ping Nie, Xiaotong Chen, and Wenhu Chen. AceCoder: Acing coder RL via automated test-case synthesis. *arXiv preprint arXiv:2502.01718*, 2025.

Xiaoqing Zhang, Yuhan Liu, Flood Sung, Xiuying Chen, Shuo Shang, and Rui Yan. Thinking before running! efficient code generation with thorough exploration and optimal refinement. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 23268–23281, 2025.

Haizhong Zheng, Yang Zhou, Brian R Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. Act only when it pays: Efficient reinforcement learning for LLM reasoning via selective rollouts. *arXiv preprint arXiv:2506.02177*, 2025.

A Notation Summary

This appendix fixes the notation used throughout the paper and, in particular, resolves three sources of ambiguity that recur in the analysis: (i) the three distinct M-quantities, (ii) the two comparators against which PROSE is measured, and (iii) the assumption labels (A1)–(A3) that enter the gradient-level results. Table 5 collects the symbols used by the method, and Table 6 those used by the analysis.

The three M-quantities. $M := N/k$ is the balanced per-plan count. It is a theoretical object: it is the per-plan count realised under uniform allocation (GRPO+CoT and the PVE stage), and it is the comparator size used in every variance statement. M_i is the realised per-plan count that PROSE produces after Stage-2 reallocation, with $\sum_{i=1}^{k} M_i = N$ and in general $M_i \neq M$. $M^{(0)}$ is the uniform Stage-1 probe budget, shared across plans, from which the plan posteriors are formed; $M^{(0)} \leq M_i$ by construction, and $M^{(0)} < M$ whenever any reallocation is to occur. At the default operating point k=4, M=4, $M^{(0)}$=2, N=16.

The two comparators. Variance and gradient statements compare PROSE's plan-conditioned quantities against a flat (grand-mean) comparator, and that comparator admits two distinct instantiations which we label consistently wherever they appear:

- **Same-batch comparator.** Both estimators are evaluated on one stored batch organised as k plans of size M. This is the quantity that is actually estimated when the flat and hierarchical advantages are recomputed on the same set of rollouts, and it is therefore the comparator that the measured variance ratio corresponds to. Writing the general one-way ANOVA identity $\mathrm{Var}(\widehat{A}^{\mathrm{flat}}_{i,j}) = \frac{k-1}{k}\sigma_B^2 + \frac{N-1}{N}\sigma_W^2$ (Appendix G) against $\mathrm{Var}(\widehat{A}_{code,i,j}) = \frac{M-1}{M}\sigma_W^2$,

$$R_{\mathrm{same}} \;=\; \frac{\frac{M-1}{M}(1-\rho_r)}{\frac{k-1}{k}\rho_r + \frac{N-1}{N}(1-\rho_r)}. \tag{6}$$

- **Deployed comparator.** GRPO as actually run draws N i.i.d. rollouts, each carrying its own latent plan. *For the flat term only* this is the degenerate case k=N, M=1 of the same identity, which reduces exactly to $\frac{N-1}{N}\sigma^2$; the plan-conditioned term it is compared against remains PROSE's, at M=N/k. Writing M for PROSE's per-plan count throughout,

$$R_{\mathrm{dep}} \;=\; \frac{(M-1)(1-\rho_r)\,N}{M\,(N-1)}. \tag{7}$$

  The ratio is cross-design by construction: the numerator carries M=4, the denominator M=1. Substituting M=1 into Equation (7) throughout would describe a comparison of GRPO against itself, and is not what this comparator means.

At PROSE's operating point (k=M=4, N=16, ρ_r=0.72) — the deployed comparator differing only in reading the flat term at k=N, M=1 — these evaluate to $R_{\mathrm{same}} = 0.262$ and $R_{\mathrm{dep}} = 0.224$, i.e. reduction fractions $1 - R$ of 0.738 and 0.776. The same-batch value is the protocol-matched one.

The deployed value answers a different, cross-design question and is never used as the reference for a measured ratio. Neither instantiation asserts that GRPO performs cluster sampling: the design effect $D_r = 1 + (M-1)\rho_r$ and the k-plan organisation are a *counterfactual clustered-sampling view*, a unit-bearing way of expressing how much of a group's reward variance is plan-level. Under i.i.d. sampling the N rollouts are marginally exchangeable and the unconditional covariance between two draws is zero; what the analysis exploits is the latent-plan mixture structure of the reward distribution, under which the flat group-relative baseline, efficient in the homogeneous case, is variance-suboptimal.

Assumption labels used by the gradient-level results. Theorem 4.1(c) and Theorem J.1 are stated under three labelled conditions, which we use with these names throughout:

- **(A1) Cross-plan score orthogonality:** $\langle h_{i,j}, h_{i',j'} \rangle = 0$ for $i \neq i'$. Only approximate in practice because of the shared problem-description prefix; Theorem H.3 gives the sign and magnitude of the relaxation, with $\widehat{\rho}_{\text{cross}} \approx 0.08$ on APPS+.
- **(A2) Non-negative within-plan score correlation:** $\rho_h^{\text{flat}} \geq 0$. The cosine proxy gives $\widehat{\rho}_h^{\text{flat}} \approx 0.56$ (Appendix L).
- **(A3) Score-norm condition:** $\|h^{\text{cond}}\|^2 \geq (M-1)\big(1-\rho_h^{\text{flat}}\big)\|h^{\text{plan}}\|^2$. This is the sufficient form; the exact requirement carries an additional factor $1 + (M-1)\rho_h^{\text{flat}}$ on the left-hand side and is therefore weaker.

At $\widehat{\rho}_h^{\text{flat}} = 0.56$ and $M = 4$, the exact form imposes a norm-ratio threshold of ≈ 0.49 versus 1.32 for the stated sufficient form. The norm ratio $\kappa := \|h^{\text{cond}}\|^2/\|h^{\text{plan}}\|^2$ that these thresholds are checked against has two proxies, and they do not agree. A proportional-to-token-count reading of the paper's norm heuristic ($\sim$35 plan tokens against $\sim$200+ code tokens) gives $\kappa \approx 5.7$, which clears every threshold above. But Equation (59) forces $(1+\kappa)\rho_h^{\text{flat}} = 1 + \kappa\,\rho_h^{\text{cond}}$, so the cosine proxies of Table 12 imply $\kappa = (1-\rho_h^{\text{flat}})/(\rho_h^{\text{flat}} - \rho_h^{\text{cond}}) \approx 1.3$ instead, and at that value the exact form still holds while the stated sufficient form (1.32) and the deployed-comparator threshold $(M-1) = 3$ do not. (A3) is thus explicit and proxy-checkable, and it is satisfied at the operating point in its exact form against the counterfactually clustered comparator; Appendix H.7 states the disagreement between the two proxies and which conclusions survive the less favourable one.

A note on the optimality class. Every optimality claim in this paper is stated *within the symmetric subtractive class* $\mathcal{F}$ of Table 6: baselines that are permutation-invariant within a plan and plan-conditionally unbiased. Estimators that exploit finer-grained signal (token-level process rewards, test-case-level partial credit, intermediate execution traces) leave $\mathcal{F}$ and are not bound by the $(M-1)\sigma_W^2/M$ floor. The UMVU statement therefore does not extend to process-level or partial-credit reward designs.

B Extended APRA Details

This appendix expands the APRA stage described in Section 4.1: the construction of the plan posterior from the Stage-1 probe, the rounding rule that turns allocation probabilities into integer counts, the interpretation of the ω_i factor in the training objective, and two optional extensions whose implementation status we state explicitly.

B.1 Posterior construction from the uniform probe

APRA deliberately avoids ϵ-greedy heuristics and derives its Stage-2 allocation from posterior plan statistics. Because execution rewards are Bernoulli, the conjugate Beta update is exact rather than an approximation: with a uniform probe of $M^{(0)}$ rollouts per plan producing s_i successes and $f_i = M^{(0)} - s_i$ failures, the posterior over the plan value v_i is $\text{Beta}(\alpha_0 + s_i,\, \beta_0 + f_i)$, whose mean and standard deviation give the μ_i and σ_i of Section 4.1. With the uniform prior $(\alpha_0, \beta_0) = (1, 1)$ and $M^{(0)} = 2$, the posterior support is coarse, with $s_i \in \{0, 1, 2\}$ yielding only three distinct (μ_i, σ_i) pairs, and this is precisely why the uncertainty term $\lambda\sigma_i$ matters: a plan with $s_i = 0$ after two probes is not treated as valueless, and the exploration floor η guarantees it retains positive allocation probability. The PVE warm-up exists for the same reason: it improves posterior calibration before the

Table 5: Method-level notation. The three M-quantities (M, M_i, $M^{(0)}$) are distinguished in the text above and should not be conflated.

Symbol	Description
Generation and reward	
x	Input problem description
p, p_i	Latent reasoning plan; i-th sampled plan, externalised as short plan text
$y, y_{i,j}$	Generated program; j-th program sampled under plan p_i
$r(y) \in \{0, 1\}$	Binary execution reward (all tests pass / not)
$r_{i,j}$	Reward of the j-th rollout under plan p_i
π_θ	Policy parameterised by θ; $\pi_{\theta_{\text{old}}}$, π_{ref} behaviour and reference policies
$J(\theta)$	Population objective $\mathbb{E}_{x,y}[r(y)]$
Budget and allocation	
N	Total execution budget per input, $\sum_{i=1}^{k} M_i = N$
k	Number of sampled plans per input
M	Balanced per-plan count N/k; theoretical comparator, and the realised count under uniform allocation
$M^{(0)}$	Stage-1 uniform probe budget per plan (APRA); $M^{(0)} \leq M_i$, and $M^{(0)} < M$ for reallocation to occur
$M_i^{(1)}$	Stage-2 reallocated count for plan p_i
M_i	Total executions allocated to plan p_i, $M_i = M^{(0)} + M_i^{(1)}$
N_{rem}	Stage-2 remaining budget $N - kM^{(0)}$
ω_i	Allocation ratio $M_i/(N/k)$; $\omega_i \equiv 1$ recovers the uniform-allocation objective
Plan posterior and allocation rule	
s_i, f_i	Stage-1 successes and failures for plan p_i; $s_i + f_i = M^{(0)}$
α_0, β_0	Beta prior parameters (method context)
α_i, β_i	Beta posterior parameters for plan p_i (not the ANOVA random effect α_i of the proof appendices)
μ_i, σ_i	Posterior mean and standard deviation for plan p_i
λ	UCB exploration coefficient
u_i	Posterior-UCB score $\mu_i + \lambda\sigma_i$
τ_{alloc}	Softmax temperature for budget allocation (not a sampling temperature)
η	Exploration floor in the allocation mixture q_i
q_i	Softmax-mixture allocation probability for plan p_i
ϕ_i, K_{ij}	Plan embedding and similarity kernel of the optional contextual transfer (Appendix B)
Advantages and objective	
$\widehat{V}_i, \bar{V}$	Empirical plan value $\frac{1}{M_i}\sum_j r_{i,j}$; grand mean $\frac{1}{k}\sum_\ell \widehat{V}_\ell$
$\widehat{A}_{plan,i}$	Plan-level advantage $\widehat{V}_i - \bar{V}$
$\widehat{A}_{code,i,j}$	Code-level (plan-conditioned) advantage $r_{i,j} - \widehat{V}_i$
$\widehat{A}_{i,j}^{\text{flat}}$	Flat grand-mean advantage $r_{i,j} - \bar{r}$ used by GRPO
$\widehat{V}_{aw}$	Allocation-weighted plan-value aggregate $\sum_i (M_i/N)\widehat{V}_i$
$\hat{r}_i^{plan}, \hat{r}_{i,j}^{code}$	Plan-level and code-level importance ratios
$\phi(\cdot, \cdot)$	GRPO clipped surrogate
β	KL penalty coefficient (code level)
ϵ_{clip}	GRPO clipping coefficient
Sampling and schedule	
$\tau_{\text{plan}}, \tau_{\text{code}}$	Plan- and code-generation sampling temperatures; **identical across all methods**
$T_{\text{PVE}}, T_{\text{APRA}}$	Number of PVE warm-up and APRA training steps

allocation signal is allowed to act, so that $\varepsilon_{\max} := \max_i |\mu_i - v_i|$ in the bias bound of Theorem F.1 is small when reallocation begins.

B.2 Integer allocation and positivity

The mixture probabilities q_i of Section 4.1 are converted into integer counts by largest-remainder rounding of $N_{\text{rem}} q_i$, where $N_{\text{rem}} = N - kM^{(0)}$. Two properties follow and are used elsewhere in the paper. First, the realised counts satisfy $\sum_i M_i^{(1)} = N_{\text{rem}}$ exactly, so the total execution budget is

Table 6: Analysis-level notation. All quantities are defined under the one-way random-effects model $r_{i,j} = v_i + \varepsilon_{i,j}$ with balanced plan size $M = N/k$.

Symbol	Description
Reward-level variance decomposition	
v_i	Plan-conditional reward mean of plan p_i (written $\mu + \alpha_i$ in the proof appendices)
$\sigma^2, \sigma_B^2, \sigma_W^2$	Total, between-plan and within-plan reward variance; $\sigma^2 = \sigma_B^2 + \sigma_W^2$
ρ_r	Within-plan reward intraclass correlation, $\rho_r := \sigma_B^2/\sigma^2 \in (0, 1)$
D_r	Counterfactual cluster design effect $1 + (M-1)\rho_r$ (unit-bearing reading of ρ_r)
N_{eff}	Effective sample size N/D_r of the counterfactual clustered mean
$R_{\text{same}}, R_{\text{dep}}$	Advantage-variance ratios against the same-batch and deployed comparators, Eqs. (6)–(7)
Gradient-level quantities	
$g, \hat{g}$	True gradient $\nabla_\theta J(\theta)$ and an unbiased estimator of it
$h_{i,j}$	Score function attached to rollout (i, j)
h_i^{plan}	Plan-level score $\nabla_\theta \log \pi_\theta(p_i \mid x)$, shared by all rollouts under plan p_i
$h_{i,j}^{\text{cond}}$	Plan-conditioned score $\nabla_\theta \log \pi_\theta(y_{i,j} \mid x, p_i)$
$h_{i,j}^{\text{flat}}$	Unconditioned score $h_i^{\text{plan}} + h_{i,j}^{\text{cond}}$ used by GRPO
$\|h^\bullet\|^2$	Expected squared score norm of component $\bullet \in \{\text{flat}, \text{cond}, \text{plan}\}$; $\|h^{\text{flat}}\|^2 = \|h^{\text{cond}}\|^2 + \|h^{\text{plan}}\|^2$ (Theorem H.2)
ρ_h^{flat}	Within-plan correlation of the unconditioned scores (GRPO); proxy $\widehat{\rho}_h^{\text{flat}} \approx 0.56$
ρ_h^{cond}	Within-plan correlation of the plan-conditioned scores (PROSE); proxy $\widehat{\rho}_h^{\text{cond}} \approx 0.21$
ρ_{cross}	Normalised cross-plan score similarity; proxy $\widehat{\rho}_{\text{cross}} \approx 0.08$
D_{eff}	Gradient design effect $1 + (M-1)\rho_h^{\text{flat}}\rho_r$ (a same-batch quantity)
$\gamma^{\text{flat}}, \gamma^{\text{cond}}, \gamma^{\text{plan}}$	Noise-to-signal ratios $\|h^\bullet\|^2\sigma^2/(N\|g\|^2)$; trajectory-uniform bounds in Theorem J.1
Σ	Gradient second moment $\mathbb{E}\|\hat{g} - g\|^2$ of an estimator
ΔJ	Per-step policy improvement at the optimised step size
Convergence and assumptions	
L, μ	Smoothness constant and Polyak–Łojasiewicz constant of J
$T_{\text{suff}}(\varepsilon)$	PL-derived sufficient iteration count $L(1 + \gamma S)\log(1/\varepsilon)/\mu$
(A1),(A2),(A3)	Cross-plan orthogonality; $\rho_h^{\text{flat}} \geq 0$; score-norm condition (see text above)
$\mathcal{F}$	Symmetric subtractive class of plan-membership advantage estimators (Theorem I.1)

N for every input and every method — budget parity across methods is enforced at the level of the allocation rule, not merely reported. Second, every plan retains $M_i \geq M^{(0)} \geq 1$ rollouts regardless of its posterior, so the plan-level advantage $\widehat{A}_{plan,i}$ is always defined and no plan is ever starved to zero support; the exploration floor η additionally keeps $q_i \geq \eta/k > 0$, which is the positivity property invoked in the bias analysis. The rounding contributes the $\frac{k}{N}(v_{\max} - v_{\min})$ term of the trivial bound in Theorem F.1.

The rule is well defined whenever $kM^{(0)} < N$, so that $N_{\text{rem}} > 0$; when $kM^{(0)} = N$ the Stage-1 probe exhausts the budget and APRA degenerates to PVE. Consequently the pair $(k, M^{(0)})$ cannot be held fixed across all execution budgets in the budget-scaling study; the per-budget instantiations are reported in Appendix R.

B.3 ALLOCATION-AWARE OBJECTIVE INTERPRETATION

Because allocation changes the empirical sampling distribution over plans, reusing an unweighted plan-level objective would bias updates toward frequently allocated plans. The factor $\omega_i = M_i/(N/k)$ in the core objective is the opposite of a standard importance-reweighting correction, which would use $1/\omega_i$ to recover the uniform-allocation objective: ω_i deliberately amplifies gradient contributions from plans in which the method has invested more budget, reinforcing the exploitation signal produced by the forward allocation. This is why the APRA stage is analysed with a bias bound rather than an unbiasedness claim: the resulting allocation-induced bias of $\widehat{V}_{aw}$ is bounded in Theorem F.1, quadratically in the plan-value spread $\widehat{\text{Var}}(v)$ and inversely in τ_{alloc}. Setting $\omega_i \equiv 1$ recovers the uniform-allocation objective exactly, which is the PVE stage and the regime in which Theorem 4.1 is stated.

B.4 Optional extension I: semantic contextual posterior transfer

Treating the k plans as independent bandit arms ignores semantic overlap among plan texts: two plans that describe the same algorithmic idea in different words receive entirely separate posteriors, and evidence gathered for one does nothing for the other. A contextual posterior removes this by sharing Stage-1 evidence through a kernel over plan embeddings.

Formulation. Let $\phi_i \in \mathbb{R}^d$ be an embedding of plan p_i, taken from the policy model's hidden states while it processes the plan text, so that feature extraction adds no forward-pass model calls beyond standard rollout prompting. Define the thresholded cosine kernel

$$K_{ij} \;=\; k(\phi_i, \phi_j) \;=\; \max\!\left(0,\; \frac{\langle \phi_i, \phi_j \rangle}{\|\phi_i\|\,\|\phi_j\|}\right) \;\in\; [0,1], \tag{8}$$

where the truncation at zero prevents negative pseudo-counts; note $K_{ii} = 1$, so a plan's own evidence always enters at full weight. Given Stage-1 counts (s_j, f_j) for every plan j, the contextual Beta posterior for plan i is formed by kernel-weighted pseudo-counts,

$$\alpha_i \;=\; \alpha_0 + \sum_{j=1}^{k} K_{ij}\, s_j, \qquad \beta_i \;=\; \beta_0 + \sum_{j=1}^{k} K_{ij}\, f_j, \tag{9}$$

and the UCB score $u_i = \mu_i + \lambda\sigma_i$ is then computed from the moments of $\mathrm{Beta}(\alpha_i, \beta_i)$ exactly as in the independent case. Equation (9) reduces to the independent-arm posterior when $K = I$, and interpolates smoothly toward a single pooled posterior as the plans become semantically indistinguishable ($K \to \mathbf{1}\mathbf{1}^\top$). Nothing downstream changes: allocation, rounding and the ω_i weighting are untouched.

A caveat on calibration. Equation (9) inflates the effective evidence count from $M^{(0)}$ to $\sum_j K_{ij}(s_j + f_j) = M^{(0)} \sum_j K_{ij}$, which is at least $M^{(0)}$ and can approach $kM^{(0)}$ when plans are near-duplicates. The posterior is therefore over-confident relative to the evidence actually collected, shrinking σ_i and weakening exactly the uncertainty term that the allocation rule relies on. A row-normalised variant, replacing K_{ij} by $K_{ij}/\sum_\ell K_{i\ell}$, preserves the total count at $M^{(0)}$ at the cost of down-weighting a plan's own observations. We do not adjudicate between the two variants here, because neither was run.

Implementation status. This extension is **a design sketch that was not enabled in any experiment reported in this paper.** Every reported PROSE, APRA and PVE number uses the independent-arm posterior, i.e. $K = I$ in Equation (9). No result, ablation, or claim in this paper depends on it, and the embedding layer from which ϕ_i would be taken is not fixed by any reported measurement. We give the full formulation so that the extension is reproducible if a reader wishes to evaluate it, but it is not part of the paper's claims.

B.5 Optional extension II: uncertainty-aware dynamic KL scaling

A second variant we considered is to let the KL penalty coefficient vary with the posterior uncertainty of the plan under which a rollout was generated, rather than holding β fixed at $1\mathrm{e}{-3}$ across all plans: the motivation is that rollouts drawn under a poorly identified plan carry less reliable advantage signal and could reasonably be held closer to the reference policy. We state this only as a named design direction. **No schedule coupling β to σ_i is specified, tuned, or evaluated anywhere in this paper, and no reported result uses anything other than a constant β.** We deliberately do not supply a formula here, since any particular choice would be an invention rather than a description of the implemented system.

C Algorithmic Details

This appendix gives the full pseudocode of PROSE: the warm-up procedure for Plan-level Value Estimation (PVE) in Algorithm 1, and the Adaptive Plan-conditioned Rollout Allocation (APRA) procedure in Algorithm 2. Both follow the same grouped-sampling training paradigm as GRPO and differ from it only in how the N rollouts per input are organised and how their advantages are computed.

C.1 STEP-BY-STEP CORRESPONDENCE WITH THE METHOD SECTION

PVE (Algorithm 1). The warm-up stage runs with $M_i = M = N/k$ and hence $\omega_i \equiv 1$, which is exactly the regime in which Theorem 4.1 is stated; every advantage computed in Algorithm 1 is unbiased, and no bias bound is needed. The plan-sampling line instantiates the factorisation $p \sim \pi_\theta(p \mid x), y \sim \pi_\theta(y \mid x, p)$ of Section 3.1, with the plan externalised as short plan text by the plan-generation prompt of Appendix S. The two advantage lines are the definitions of $\widehat{A}_{plan,i} = \widehat{V}_i - \bar{V}$ and $\widehat{A}_{code,i,j} = r_{i,j} - \widehat{V}_i$ given in Section 4.1; the second is the quantity whose variance $\frac{M-1}{M}(1-\rho_r)\sigma^2$ appears in Theorem 4.1(a) and which attains the UMVU floor within the symmetric subtractive class in Theorem 4.1(b). The final update line applies the core objective of Section 4.1 with $\omega_i \equiv 1$, in which form it coincides with the uniform-allocation objective.

APRA (Algorithm 2). Stage 1 allocates $M^{(0)}$ rollouts per plan uniformly and forms the Beta posterior described in Appendix B; the UCB line computes $u_i = \mu_i + \lambda\sigma_i$. Stage 2 evaluates the softmax mixture q_i of Section 4.1 at temperature τ_{alloc} with exploration floor η, and converts it to integer counts $M_i^{(1)}$ by largest-remainder rounding, giving $M_i = M^{(0)} + M_i^{(1)}$ and $\omega_i = M_i/(N/k)$. All M_i rewards, probe and reallocated together, are pooled before $\widehat{V}_i$ is recomputed, so the plan value uses the full realised budget rather than the probe alone. The stored tuple carries ω_i alongside the two advantages because the plan-level term of the objective is ω_i-weighted while the code-level term is not; this asymmetry is the content of the ω_i discussion in Appendix B, and it is the source of the allocation-induced bias bounded by Theorem F.1.

Table 7 lists the correspondence explicitly.

Table 7: Correspondence between pseudocode steps and the definitions of Section 4.1. "Unbiased" refers to plan-conditional unbiasedness of the advantage.

Pseudocode step	**Defined in**	**Status**
Sample $\{p_i\}_{i=1}^k \sim \pi_\theta(\cdot \mid x)$	Section 3.1	—
Sample $y_{i,j} \sim \pi_\theta(\cdot \mid x, p_i)$	Section 3.1	—
Beta posterior (α_i, β_i) from (s_i, f_i)	Section 4.1, Appendix B	exact (Bernoulli conjugacy)
UCB score $u_i = \mu_i + \lambda\sigma_i$	Section 4.1	—
Mixture q_i at τ_{alloc}, floor η	Section 4.1	—
Largest-remainder rounding to $M_i^{(1)}$	Appendix B	budget-exact, positivity-preserving
$\widehat{V}_i = \frac{1}{M_i}\sum_j r_{i,j}$, $\bar{V} = \frac{1}{k}\sum_\ell \widehat{V}_\ell$	Section 4.1	unbiased at $\omega_i \equiv 1$
$\widehat{A}_{plan,i} = \widehat{V}_i - \bar{V}$	Section 4.1	unbiased at $\omega_i \equiv 1$
$\widehat{A}_{code,i,j} = r_{i,j} - \widehat{V}_i$	Section 4.1	UMVU in $\mathcal{F}$ (Theorem I.1)
$\omega_i = M_i/(N/k)$ weighting in the objective	Section 4.1	biased; bound in Theorem F.1

C.2 SAMPLING TEMPERATURES

Three temperatures appear in the algorithms and hyperparameter tables and denote different objects. τ_{plan} and τ_{code} are *sampling* temperatures, for plan text and for program text respectively; both are set to the same value and are **standardised across all methods (Base, GRPO, GRPO+CoT, PVE, APRA and PROSE) for both training rollouts and evaluation decoding**, together with top-$p = 1.0$ (Table 29). No method in this paper is generated at a temperature different from any other; differences in pass@1 are therefore not attributable to a decoding confound. τ_{alloc} is not a sampling temperature at all: it is the softmax temperature of the allocation mixture over plan UCB scores, it applies only to APRA's Stage 2, and it appears in the bias bound of Theorem F.1 through the high-temperature condition $c := \max_i |u_i - \bar{u}|/\tau_{\text{alloc}} < 1$.

C.3 COST AND IMPLEMENTATION FOOTPRINT

Neither algorithm adds execution cost: the total number of executed programs per input is N for PROSE exactly as for GRPO, and the only additional generation is the plan text itself (∼35–40 tokens per problem, at or below the ∼5–8% token class of GRPO+CoT; the aggregate measured overhead is ≈3%, Appendix O.9). The additional computation (k Beta posterior updates, k UCB

scores, one softmax and one largest-remainder rounding per input) is $O(k)$ arithmetic and negligible beside rollout generation and sandboxed execution. As an engineering estimate the additional code is small and the runtime cost negligible against LLM inference; Appendix R gives the line counts, the recommended minimal entry point, and the measured wall-clock overhead.

Algorithm 1 PVE Warm-up: Plan-level Value Estimation

Input: training batch $\mathcal{B}$, policy π_θ, reward function $r(\cdot)$
Hyperparameters: number of plans k, balanced executions per plan $M{=}N/k$, warm-up steps T_{PVE}
for $t = 1$ **to** T_{PVE} **do**
  Initialize rollout buffer $\mathcal{D} \leftarrow \emptyset$
  for each input $x \in \mathcal{B}$ **do**
    Sample plans $\{p_i\}_{i=1}^{k} \sim \pi_\theta(\cdot \mid x)$
    Sample programs $\{y_{i,j} \sim \pi_\theta(\cdot \mid x, p_i)\}_{i=1,j=1}^{k,M}$
    Execute the $\{y_{i,j}\}_{i=1,j=1}^{k,M}$ to obtain rewards $\{r_{i,j}\}$
    Compute plan values $\widehat{V}_i \leftarrow \frac{1}{M}\sum_{j=1}^{M} r_{i,j}$ for $i = 1, \ldots, k$
    $\bar{V} \leftarrow \frac{1}{k}\sum_{\ell=1}^{k} \widehat{V}_\ell$
    for each plan p_i **do**
      $\widehat{A}_i^{plan} \leftarrow \widehat{V}_i - \bar{V}$
      for $j = 1$ **to** M **do**
        $\widehat{A}_{i,j}^{code} \leftarrow r_{i,j} - \widehat{V}_i$
        Store $(x, p_i, y_{i,j}, \widehat{A}_i^{plan}, \widehat{A}_{i,j}^{code})$ into $\mathcal{D}$
      end for
    end for
  end for
  Update π_θ using $\mathcal{L}_{\text{PVE}}(\theta)$ with buffer $\mathcal{D}$ // core objective at $\omega_i \equiv 1$
end for

Algorithm 2 APRA: Uncertainty-aware Plan-conditioned Rollout Allocation

Input: batch $\mathcal{B}$, policy π_θ, reward $r(\cdot)$
Hyper: plans k, Stage-1 probe $M^{(0)}$ (requires $kM^{(0)} < N$), total budget N, prior (α_0, β_0), UCB λ, floor η, temperature τ_{alloc}
for $t = 1$ **to** T_{APRA} **do**
  Initialize $\mathcal{D} \leftarrow \emptyset$
  for each input $x \in \mathcal{B}$ **do**
    Sample plans $\{p_i\}_{i=1}^{k} \sim \pi_\theta(\cdot \mid x)$
    // Stage 1: Uniform exploration — sample $M^{(0)}$ programs per plan, execute to get $\{r_{i,j}^{(0)}\}_{j=1}^{M^{(0)}}$
    Beta posteriors: $(\alpha_i, \beta_i) \leftarrow (\alpha_0 + s_i,\ \beta_0 + M^{(0)} - s_i)$ where $s_i = \sum_j r_{i,j}^{(0)}$; derive (μ_i, σ_i)
    UCB score: $u_i \leftarrow \mu_i + \lambda\sigma_i$
    // Stage 2: Adaptive exploitation — allocate remaining $N_{\text{rem}} = N - kM^{(0)}$ samples
    $q_i \ \leftarrow\ (1-\eta)\,\text{softmax}(u_i/\tau_{\text{alloc}}) + \eta/k$; allocate $M_i^{(1)}$ from $\{q_i\}$ by largest-remainder rounding s.t. $\sum_i M_i^{(1)} = N_{\text{rem}}$
    Execute $M_i^{(1)}$ additional samples; set $M_i \leftarrow M^{(0)} + M_i^{(1)}$, pool all rewards $\{r_{i,j}\}_{j=1}^{M_i}$
    $\widehat{V}_i \leftarrow \frac{1}{M_i}\sum_j r_{i,j},\ \bar{V} \leftarrow \frac{1}{k}\sum_\ell \widehat{V}_\ell,\ \omega_i \leftarrow M_i/(N/k)$
    for each $(p_i, y_{i,j})$ **do**
      Store $(x, p_i, y_{i,j}, \omega_i,\ \widehat{V}_i - \bar{V},\ r_{i,j} - \widehat{V}_i)$ into $\mathcal{D}$
    end for
  end for
  Update π_θ via $\mathcal{L}_{\text{APRA}}(\theta)$ with $\mathcal{D}$ // core objective with ω_i-weighted plan term
end for

Full schedule. A PROSE run is Algorithm 1 for T_{PVE} steps followed by Algorithm 2 for T_{APRA} steps from the resulting checkpoint, for a total matching the optimisation-step budget of every baseline. The warm-up is what makes $\varepsilon_{\max}$ small enough for the refined bias bound of Theorem F.1 to be informative when reallocation begins; removing it is the failure mode that motivates the two-stage design.

D EXTENDED PROBLEM FORMULATION

This appendix carries the material trimmed from Section 3 for length, in the order in which the main text points to it: the motivation for the latent-plan factorization and why the plan is the right granularity for credit assignment (Appendix D.1); the marginal-versus-conditional bookkeeping that the compressed main-text statement summarizes (Appendix D.2); the extended reading of the design-effect diagnostic (Appendix D.3); and the gradient-level route (Appendix D.4). The numerical instantiations of the advantage-variance identity are quoted below only where the argument needs them; they are stated in Theorem 4.1(a) and derived in Appendix G.3.

D.1 WHY THE PLAN IS THE CREDIT-ASSIGNMENT GRANULARITY

The latent plan $p \in \mathcal{P}$ of Section 3.1 ranges over the decisions that fix a solution's algorithmic identity before any code is written: how the problem is decomposed, which algorithm or algorithmic family is chosen, and which data structures carry the state. It is externalized as a short textual plan by the planning prompt of Appendix S ($\approx$35–40 tokens per problem), so that plan identity is observable at training time rather than inferred.

Two consequences of this factorization motivate the rest of the paper.

Programs sharing a plan differ only in low-level implementation detail. Their execution outcomes are therefore strongly coupled through the plan they realize: correct implementations of the same plan typically succeed together, whereas a flawed plan uniformly produces incorrect programs regardless of how carefully the low-level details are written. A rollout's success probability is thus largely determined by its plan rather than by token-level sampling noise; that is the qualitative content of a reward ICC ρ_r bounded away from zero, measured at 0.72 ± 0.14 on APPS+ and 0.48 ± 0.16 on GSM8K (Appendix K).

The plan is therefore also the right granularity for credit assignment. Under a sparse binary execution reward the learning signal available to a flat group-relative baseline is a single scalar per rollout, and that scalar mostly reports which plan the rollout realized. Attributing it at rollout granularity spends the comparison on implementation noise; attributing it at plan granularity (one plan-level contrast across plans, one code-level contrast within a plan) separates the two sources *a priori* rather than estimating their mixture. This is the structural reason the hierarchical advantage of Section 4.1 is the natural estimator here, and Theorem 4.1 is the formal version of the same statement.

We stress that the factorization is a modelling device, not a claim about the policy's internal computation. Nothing in the analysis requires that the model "plans" in any mechanistic sense; it requires only that the externalized plan text is informative enough about the eventual outcome for $\rho_r > 0$ to hold, which is an empirical question answered in Appendix K. Where plans are not distinguishable (near-duplicate plans within a problem), the mechanism has correspondingly less to work with, and the gain shrinks accordingly (Appendix O).

D.2 MARGINAL VERSUS CONDITIONAL BOOKKEEPING

The main-text statement compresses a bookkeeping point worth spelling out, because it is the point at which the paper's diagnosis was originally mis-stated.

GRPO draws its N rollouts i.i.d. from the plan-marginal $\pi_\theta(y \mid x) = \sum_p \pi_\theta(p \mid x)\, \pi_\theta(y \mid x, p)$. At that marginal scale everything is as it should be: the N draws are marginally exchangeable, the unconditional covariance between two draws is exactly zero, and the group mean has variance $\mathrm{Var}(\bar{r}) = \sigma^2/N$ exactly, with no hidden loss of effective sample size. Simultaneously, two rollouts that happen to realize the *same* plan have covariance $\sigma_B^2 > 0$. These two facts are not in tension: the first is a statement about independently sampled plans, the second a statement conditional on

plan membership, and both hold under i.i.d. sampling of a mixture. What follows from the second is not a defect of the sampler but a property of the reward distribution, and a group-relative baseline, which uses the same N rollouts as both target and baseline, is exactly the object that has to contend with it. The efficiency question is therefore a question about the *baseline*, not about the sampling: the flat grand mean is the efficient subtractive control variate in the homogeneous case $\rho_r = 0$ and is variance-suboptimal once $\rho_r > 0$, where the efficient choice is the plan-conditioned $\widehat{V}_i$ — the complete sufficient statistic for v_i under Equation (1), which is what lets Theorem I.1 apply Lehmann–Scheffé directly. Appendix K.2 states the same scope restriction from the measurement side.

D.3 The design-effect diagnostic, in full

Equation (2) translates ρ_r into effective-sample-size units, which are easier to reason about than a raw correlation, at the cost of invoking a sampling design that GRPO does not use. Three points complete the main-text statement.

Why the counterfactual is nonetheless the operative one. Clustered organization is exactly what PROSE adopts: arranging a batch as k plans of M rollouts is what makes the plan-conditioned baseline available in the first place. D_r therefore measures how much plan-level structure that reorganization has to work with; it is a statement about the opportunity, not about a loss GRPO is currently incurring at the sampling stage. At $\rho_r{\approx}0.72$ and $M{=}4$ this gives $D_r \approx 3.16$ and $N_{\text{eff}}/N \approx 0.32$; on GSM8K, $\rho_r{\approx}0.48$ gives $D \approx 2.44$ and $N_{\text{eff}}/N \approx 0.41$ (Table 11). A random-grouping control, which assigns rollouts to groups without regard to plan, gives $\rho_r \approx 0.05$: the structure is a property of plan membership, not of grouping per se.

What D_r is not. It is not the variance-reduction factor. The reduction fractions the paper claims come from the advantage-level identity of Theorem 4.1(a) — 0.738 against the same-batch flat comparator and 0.776 against GRPO as deployed, at $k{=}M{=}4$, $N{=}16$, $\rho_r{=}0.72$ — and it is that identity, not D_r, that the direct measurement of Appendix K.4 tests. D_r is also not a variance-inflation claim about GRPO's i.i.d. group mean, whose variance is σ^2/N exactly; and it is not a claim that GRPO performs two-stage cluster sampling.

Where the diagnostic is used. D_r appears in the paper only in its diagnostic role: as the quantity that orders domains by expected benefit (code 3.16 versus math 2.44, matching the ordering of the observed gains) and as the quantity that justifies clustering at plan granularity in the first place. Cross-domain ordering is the only test the data support, since the gradient-level correlation ρ_h^{flat} was not measured on GSM8K (Appendix K).

D.4 The gradient-level route

The same latent structure acts a second time one level up. The within-plan score correlation ρ_h^{flat} couples multiplicatively to ρ_r, producing the within-method gradient design effect $D_{\text{eff}} = 1 + (M{-}1)\rho_h^{\text{flat}}\rho_r$ of Theorem 4.1(c), which at $\widehat{\rho}_h^{\text{flat}} = 0.56$ and $\rho_r = 0.72$ with $M{=}4$ is ≈ 2.21 (Appendices H and L). Like D_r, it is a same-batch, counterfactually clustered quantity; the as-deployed analogue has $D_{\text{eff}} \to 1$, and the two comparators are labelled as such wherever they appear.

The main text reads this route as design rationale rather than as a measured prediction, for one reason: $\widehat{\rho}_h^{\text{flat}}$ is a last-layer hidden-state cosine proxy (0.56 ± 0.09, against $\widehat{\rho}_h^{\text{cond}} = 0.21 \pm 0.06$ and $\widehat{\rho}_{\text{cross}} = 0.08 \pm 0.03$), not a direct measurement of a score-function correlation, and its dispersion propagates straight into D_{eff}. Translating D_{eff} into a pass@1 increment through the measured budget curve is accordingly presented as a secondary sanity check and never as a theorem-level prediction of accuracy; the paper's direct quantitative test is the advantage-variance ratio (0.262 predicted against 0.26 measured, Section 5). What the route does establish is which measurable quantity says when the mechanism should help, and that the gradient-level statement degrades gracefully: it rests on (A1)–(A3), of which (A1) is approximate at $\widehat{\rho}_{\text{cross}} \approx 0.08$ and (A3) is proxy-checked with a $\geq 4\times$ margin (Appendix L.2). PROSE addresses both routes with one mechanism: it estimates plan values at the correct cluster granularity, and it reallocates execution budget across rather than within plans (Section 4).

E Extended Method Details

This appendix holds material that Section 4 states in compressed form: the full disambiguation of the three per-plan counts, the design rationale for the allocation weight ω_i, the core objective written out, the split-half protocol behind the variance-ratio check, the positioning table against related GRPO variants, and the computational footprint of the posterior machinery. Nothing here is new evidence; it is the long form of what the main text asserts.

E.1 The three per-plan counts

Three M-quantities appear throughout the paper and are easy to conflate, so we separate them explicitly (see also Appendix A).

- $M := N/k$ is the *balanced* per-plan count. It is used only as the theoretical comparator against uniform-allocation GRPO: every statement of Theorem 4.1 is made at $M_i = M$, $\omega_i \equiv 1$, which is exactly the PVE stage. M is not a quantity PROSE realizes once APRA is active.
- M_i is the data-dependent count PROSE actually realizes for plan i after Stage-2 reallocation, with $\sum_i M_i = N$ and generally $M_i \neq M$. All APRA-stage estimators ($\widehat{V}_i$, ω_i) are formed at M_i, not at M.
- $M^{(0)} \leq M_i$ is the uniform Stage-1 probe budget, spent identically on every plan before any posterior is formed. It must satisfy $kM^{(0)} < N$ so that Stage 2 has budget left to allocate (Algorithm 2).

The distinction matters for reading Theorem 4.1(a): the reduction fractions 0.738 (same-batch) and 0.776 (as-deployed) are stated at the balanced comparator $k{=}M{=}4$, $N{=}16$, and therefore describe the PVE stage. The APRA stage trades a bounded bias (Theorem F.1) for the reallocation, and its net benefit is an empirical quantity, as Table 1 records.

E.2 Why ω_i is not an importance correction

The realized allocation ratio $\omega_i = M_i/(N/k)$ enters the plan-level term of Equation (4) multiplicatively. The natural alternative is the importance correction $1/\omega_i$, which would reweight each plan back to the uniform-allocation measure and so restore the uniform-allocation objective in expectation. We deliberately do not do this. Dividing by ω_i would cancel precisely the signal the forward allocation produced: plans that the posterior judged promising receive more executions, and an importance correction would erase that judgement from the backward pass, leaving the allocation mechanism with no effect on the gradient at all.

Multiplying by ω_i instead aligns backward credit strength with forward budget commitment. This is a biased estimator of the uniform-allocation gradient by construction, which is why the bias is bounded rather than assumed away: Theorem F.1 shows the bias is at most quadratic in the plan-value spread and that the per-step improvement remains positive once warm-up has calibrated the posterior. The exploration floor η in Equation (3) is what keeps that bound finite — it holds every allocation probability away from zero, preventing collapse onto a single plan and keeping ω_i bounded above.

E.3 The core objective, written out

The two policy ratios and the KL term of Equation (4) are

$$\hat{r}_i^{\text{plan}} = \frac{\pi_\theta(p_i \mid x)}{\pi_{\theta_{\text{old}}}(p_i \mid x)}, \qquad \hat{r}_{i,j}^{\text{code}} = \frac{\pi_\theta(y_{i,j} \mid x, p_i)}{\pi_{\theta_{\text{old}}}(y_{i,j} \mid x, p_i)}, \qquad \text{KL}_i = \text{KL}\big[\pi_\theta(\cdot \mid x, p_i) \,\|\, \pi_{\text{ref}}(\cdot \mid x, p_i)\big].$$

KL regularization is applied at the code level only. Plan-level drift is instead controlled by the clipping in $\phi(\cdot, \cdot)$ acting on $\hat{r}_i^{\text{plan}}$ and by the exploration floor η, which keeps the plan distribution from collapsing onto whichever plan the posterior currently favours. The ω_i-weighted plan term and the unweighted code term therefore respond to the allocation differently: only the plan level is scaled by the budget a plan received.

E.4 SPLIT-HALF PROTOCOL FOR THE VARIANCE-RATIO CHECK

The advantage-variance ratio reported in Section 5 is obtained by recomputing both the flat and the plan-conditioned advantage on the *same* stored rollouts of the k=4, M=4, N=16 batch, and comparing against the same-batch prediction $R_{\text{same}} = 0.262$. The comparison has no free parameter: ρ_r enters from the independent ICC measurement of Appendix K and (k, M, N) from the training configuration.

Two caveats attach to it. First, the measured 0.26 carries two significant figures and no reported standard error, so the agreement is an internal-consistency check on the random-effects specification, not a precision claim. Second, ρ_r and the recomputed ratio come from the same pool of problems, so the check as stated is in-sample. To rule out that the agreement is an artifact of fitting and evaluating on the same data, we specify — but have not yet run — a split-half version: ρ_r is fit on a random half of the held-out problems, the ratio is predicted from that estimate alone, and the prediction is evaluated on the disjoint half (Appendices G.5 and K.4). The deployed instantiation $R_{\text{dep}} = 0.224$ is reported separately and labelled as the cross-design quantity it is, so that the two comparators are never paired against a single measurement.

E.5 POSITIONING AGAINST RELATED GRPO VARIANTS

Table 8 places PROSE against the most closely related methods on the three design axes of Section 4. The pattern is that prior adaptive-budget work allocates on difficulty or reward density without any plan structure, prior plan-structured work does not adapt the budget, and no prior method makes the training objective aware of the allocation it performed.

Table 8: PROSE against the most closely related methods on the three design axes of Section 4, plus training-time token overhead per input relative to GRPO (measured against average generated code length). Token overhead is a generation cost, distinct from execution budget and wall-clock (Table 4). The ∼5–8% figures are the estimated class for plan-generating methods; where we have aggregate training-log measurements they are $\approx 3\%$ for PROSE (6.3k generated tokens per problem) and $\approx 7\%$ for GRPO+CoT (6.5k), against vanilla GRPO's 6.1k (Appendix O.9).

Method	Plan Structure	Adaptive Budget	Allocation-Aware Obj.	Token Overhead
GRPO (Guo et al., 2025)	✗	✗	✗	0% (baseline)
GRPO + CoT (plan prompt)	✓	✗	✗	∼5–8%
PLANSEARCH (Lei et al., 2025)	✓	✗	✗(inference only)	0% (train-time)
REINFORCE-ADA (Xiong et al., 2025)	✗	✓(difficulty)	✗	0%
XRPO (Bamba et al., 2025)	✗	✓(reward density)	✗	0%
Not-All-Rollouts (Xu et al., 2025)	✗	✓(down-sampling)	✗	0%
PRLCoder (Ye et al., 2025)	✗	✗	✗(process reward)	∼10–15% (PRM)
PROSE (ours)	✓	✓(posterior-UCB)	✓(ω_i weighting)	∼5–8% (measured: ∼3%)

E.6 COMPUTATIONAL FOOTPRINT OF THE POSTERIOR MACHINERY

Beyond the line counts reported in Section 4, the per-input arithmetic PROSE adds is $O(k)$: k Beta posterior updates, k UCB scores, one softmax and one largest-remainder rounding, all closed-form on k=4 scalars. No additional backward pass is introduced, and the two-stage schedule is a single configuration switch at step 500 rather than a separate training phase with its own optimizer state. Wall-clock accounting is in Table 4.

F PROOF OF PROPOSITION F.1 (ALLOCATION-AWARE ESTIMATION)

This appendix bounds the bias that PROSE deliberately introduces when it replaces the uniform per-plan budget by the posterior-coupled allocation of Eq. (3). The analysis is carried out in the two-stage probability model that APRA (Algorithm 2) actually realizes: the Stage-2 counts are functions of the Stage-1 probe outcomes, and the Stage-1 rewards are themselves part of the pooled plan-value estimates. No step below conditions on the realized counts $\{M_i\}$; every statement is either pathwise in the Stage-1 randomness or taken under the full joint law, and the two are combined by the law of total expectation at the end. Remark F.2 exhibits a two-plan instance in which conditioning on $\{M_i\}$ is genuinely invalid, which is why the argument is organised this way.

Two-stage probability model. Fix an input x and a plan set $\{p_i\}_{i=1}^k$ with (unknown, deterministic) success probabilities $v_i := \Pr[r = 1 \mid x, p_i]$, and write

$$\bar{v} := \frac{1}{k}\sum_{i=1}^{k} v_i, \qquad \widehat{\mathrm{Var}}(v) := \frac{1}{k}\sum_{i=1}^{k}(v_i - \bar{v})^2, \qquad \Delta_v := v_{\max} - v_{\min}. \tag{10}$$

Note that $\bar{v}$, $\widehat{\mathrm{Var}}(v)$ and Δ_v are deterministic: they are functionals of the true plan values, not of any sample. The sampling proceeds as follows.

1. **Stage 1 (uniform probe).** Each plan receives a deterministic budget of $M^{(0)}$ rollouts with rewards $r_{i,j} \sim \mathrm{Bernoulli}(v_i)$, independent across i and j. Let $\mathcal{F}_1 := \sigma\big(\{r_{i,j}\}_{i\le k,\, j\le M^{(0)}}\big)$ be the Stage-1 σ-field and $s_i := \sum_{j\le M^{(0)}} r_{i,j}$.
2. **Allocation (an $\mathcal{F}_1$-measurable decision).** The Beta–Bernoulli posterior of plan i has parameters $(\alpha_0 + s_i, \beta_0 + M^{(0)} - s_i)$ with $\alpha_0 = \beta_0 = 1$, hence mean μ_i and standard deviation σ_i with $\alpha_i + \beta_i = M^{(0)} + 2$; the posterior score is $u_i = \mu_i + \lambda\sigma_i$ and the allocation probabilities are $q_i = (1-\eta)\tilde{q}_i + \eta/k$ with $\tilde{q}_i \propto \exp(u_i/\tau_{\mathrm{alloc}})$. Integer counts are obtained by largest-remainder rounding. Every one of $\mu_i, \sigma_i, u_i, q_i, M_i$ is $\mathcal{F}_1$-measurable, and in particular none of them is independent of the Stage-1 rewards that enter $\widehat{V}_i$.
3. **Stage 2 (adaptive exploitation).** The remaining rollouts are drawn with $r_{i,j} \mid \mathcal{F}_1 \sim \mathrm{Bernoulli}(v_i)$, independent across rollouts and independent of Stage 1 given the plan index.

Writing M_i for the realized total count of plan i (so $\sum_i M_i = N$), we use throughout

$$\widehat{V}_i = \frac{1}{M_i}\sum_{j=1}^{M_i} r_{i,j}, \qquad \widehat{V}_{aw} = \sum_{i=1}^{k}\frac{M_i}{N}\widehat{V}_i, \qquad w_i := \frac{M_i}{N}, \qquad V_M := \sum_{i=1}^{k} w_i v_i, \tag{11}$$

with the convention $w_i\widehat{V}_i := 0$ when $M_i = 0$, so that $\widehat{V}_{aw}$ is well defined even if a plan receives no rollouts (under the deployed scheme $M_i \ge M^{(0)} \ge 1$, so this never occurs). Two points deserve emphasis before the proof. First, V_M is a *random* target: it is the allocation objective that the realized budget split defines, and it is $\mathcal{F}_1$-measurable. Second, $\widehat{V}_{aw}$ is the pooled success rate over all N executed rollouts,

$$\widehat{V}_{aw} = \sum_{i=1}^{k}\frac{M_i}{N}\cdot\frac{1}{M_i}\sum_{j=1}^{M_i} r_{i,j} = \frac{1}{N}\sum_{i=1}^{k}\sum_{j=1}^{M_i} r_{i,j}, \tag{12}$$

a fact we exploit repeatedly: the M_i cancel, so $\widehat{V}_{aw}$ never divides by a random count.

Proposition F.1 (Allocation-aware estimation under softmax-mixture allocation)**.** *Consider the two-stage model above, with $\varepsilon_{\max} := \max_i |\mu_i - v_i|$ and $c := \max_i |u_i - \bar{u}|/\tau_{\mathrm{alloc}}$ (both $\mathcal{F}_1$-measurable random variables, $\bar{u} = \frac{1}{k}\sum_i u_i$). Then:*

(i) (Unbiasedness for the allocated objective.) $\mathbb{E}\big[\widehat{V}_{aw}\big] = \mathbb{E}[V_M] = \sum_{i=1}^{k}\frac{\mathbb{E}[M_i]}{N} v_i$, *with no independence assumption between $\{M_i\}$ and the rewards.*

(ii) (Sharpened error ceiling.) $\mathbb{E}\big[(\widehat{V}_{aw} - V_M)^2\big] = \frac{1}{N}\,\mathbb{E}\big[\sum_i w_i v_i(1-v_i)\big] \le \frac{\mathbb{E}[V_M]\big(1-\mathbb{E}[V_M]\big)}{N} \le \frac{1}{4N}$, *the last inequality strict whenever $\mathbb{E}[V_M] \ne 1/2$.*

(iii) (Trivial bias bound, any temperature.) *For every $\tau_{\mathrm{alloc}} > 0$, $\big|\mathbb{E}[\widehat{V}_{aw}] - \bar{v}\big| \le (1-\eta)\frac{k-1}{k}\Delta_v + \frac{k}{4N}\Delta_v$.*

(iv) (Refined bias bound in the high-temperature regime.) *If in addition $c < 1$ almost surely, then*

$$\big|\mathbb{E}[\widehat{V}_{aw}] - \bar{v}\big| \le \tfrac{1-\eta}{\tau_{\mathrm{alloc}}}\left(\widehat{\mathrm{Var}}(v) + \big(\mathbb{E}[\varepsilon_{\max}] + \tfrac{\lambda}{2\sqrt{M^{(0)}+3}}\big)\sqrt{\widehat{\mathrm{Var}}(v)}\right) + \mathcal{O}\big(\mathbb{E}[c^2]\,\Delta_v\big) + \tfrac{k}{4N}\Delta_v, \tag{13}$$

where the $\mathcal{O}(\cdot)$ term is bounded explicitly by $\frac{1-\eta}{2}\mathbb{E}[c^2]\Delta_v + \mathcal{O}(\mathbb{E}[c^3]\Delta_v)$.

In particular the leading bias term is quadratic in the plan-value spread; the two remaining terms are linear in the spread but carry the small prefactors $\mathbb{E}[\varepsilon_{\max}]/\tau_{\mathrm{alloc}}$ *and* $k/(4N)$.

The bound follows from four individually controlled error sources: unbiasedness relative to the realized allocation together with the error ceiling (Part 1), the rounding mismatch between integer counts and the real-valued allocation (Part 2), the softmax bias at finite temperature (Part 3), and the calibration error of the posterior score (Part 4). Part 5 assembles them under the law of total expectation. Parts 2–4 are *pathwise* statements: they hold for every realization of $\mathcal{F}_1$, which is what allows the outer expectation to be taken at the end without any independence assumption.

Why the argument cannot condition on the realized counts. A shorter derivation would condition on $\{M_i\}$, assert $\mathbb{E}[\widehat{V}_{aw} \mid \{M_i\}] = V_M$, and take expectations. That step is not valid here. The counts are functions of the Stage-1 rewards, and those same rewards are pooled into $\widehat{V}_i$; conditioning on a large M_i therefore tilts the conditional law of the Stage-1 portion of $\widehat{V}_i$ upward — the selection effect familiar from adaptive data collection (Russo et al., 2018). The following instance makes the failure explicit.

Remark F.2 (Conditioning on $\{M_i\}$ is invalid; the unconditional statement survives). Take $k = 2$, $M^{(0)} = 1$, $N = 4$, $\eta = 0$ and $\tau_{\mathrm{alloc}} \to 0$, so that Stage 2 awards both remaining rollouts to the plan with the strictly larger probe reward and splits them one–one on a tie. Suppose $v_1 = v_2 = v$. The event $\{M_1 = 3\}$ is exactly $\{r_{1,1} = 1,\ r_{2,1} = 0\}$, so by (12)

$$\mathbb{E}\big[\widehat{V}_{aw} \,\big|\, M_1 = 3\big] = \tfrac{1}{4}\big(1 + 2v + 0\big) = \tfrac{1+2v}{4}, \qquad \text{whereas} \qquad V_M = \tfrac{3}{4}v + \tfrac{1}{4}v = v, \tag{14}$$

a discrepancy of $(1 - 2v)/4$, which vanishes only at $v = 1/2$. The conditional identity is thus false, and the conditional-variance computation built on it fails for the same reason: given $\{M_i\}$, the Stage-1 rewards are no longer $\mathrm{Bernoulli}(v_i)$. The *unconditional* conclusion nevertheless holds (here every one of the four rollouts has unconditional mean v, so $\mathbb{E}[\widehat{V}_{aw}] = v = \mathbb{E}[V_M]$), and Theorem F.3 recovers it in general by filtering forward in execution order rather than conditioning on the outcome of the allocation.

Part 1: unbiasedness for the allocated objective and the sharpened error ceiling. Index the N executed rollouts in execution order $t = 1, \ldots, N$ (all $kM^{(0)}$ Stage-1 rollouts first, then the Stage-2 rollouts in any order fixed by the implementation), let $a_t \in \{1, \ldots, k\}$ be the plan of the t-th rollout and r_t its reward, and let $\mathcal{G}_t := \sigma(a_1, r_1, \ldots, a_t, r_t)$. The scheme has two properties, and only these two are used:

- **Predictability.** a_t is $\mathcal{G}_{t-1}$-measurable. For $t \le kM^{(0)}$ it is deterministic; for $t > kM^{(0)}$ it is a function of $\mathcal{F}_1 \subseteq \mathcal{G}_{t-1}$ through $\{q_i\}$ and the rounding rule.
- **Conditional Bernoulli rewards.** $\mathbb{E}[r_t \mid \mathcal{G}_{t-1}] = v_{a_t}$ and $\mathrm{Var}(r_t \mid \mathcal{G}_{t-1}) = v_{a_t}(1 - v_{a_t})$.

Lemma F.3 (Predictable allocation)**.** *Under predictability and conditional Bernoulli rewards, with* $\widehat{V}_{aw} = \frac{1}{N}\sum_{t=1}^{N} r_t$ *and* $V_M = \frac{1}{N}\sum_{t=1}^{N} v_{a_t} = \sum_i w_i v_i$,

$$\mathbb{E}\big[\widehat{V}_{aw} - V_M\big] = 0, \quad \mathbb{E}\big[(\widehat{V}_{aw} - V_M)^2\big] = \frac{1}{N^2}\,\mathbb{E}\left[\sum_{t=1}^{N} v_{a_t}(1 - v_{a_t})\right] = \frac{1}{N}\,\mathbb{E}\left[\sum_{i=1}^{k} w_i\, v_i(1 - v_i)\right]. \tag{15}$$

Proof. Set $\xi_t := r_t - v_{a_t}$. Predictability makes v_{a_t} $\mathcal{G}_{t-1}$-measurable, so $\mathbb{E}[\xi_t \mid \mathcal{G}_{t-1}] = 0$: $\{\xi_t\}$ is a martingale difference sequence for $\{\mathcal{G}_t\}$, and $N(\widehat{V}_{aw} - V_M) = \sum_t \xi_t$ by (12). Taking expectations gives the first claim. For the second, for $s < t$, $\mathbb{E}[\xi_s \xi_t] = \mathbb{E}\big[\xi_s\, \mathbb{E}[\xi_t \mid \mathcal{G}_{t-1}]\big] = 0$, so the cross terms vanish and $\mathbb{E}[(\sum_t \xi_t)^2] = \sum_t \mathbb{E}[\xi_t^2] = \sum_t \mathbb{E}\big[\mathrm{Var}(r_t \mid \mathcal{G}_{t-1})\big] = \mathbb{E}\big[\sum_t v_{a_t}(1 - v_{a_t})\big]$. Collecting the t with $a_t = i$ gives $\sum_t v_{a_t}(1 - v_{a_t}) = \sum_i M_i v_i(1 - v_i) = N \sum_i w_i v_i(1 - v_i)$. □

Theorem F.3 delivers Proposition F.1(i) directly, in the form $\mathbb{E}[\widehat{V}_{aw}] = \mathbb{E}[V_M] = \sum_i (\mathbb{E}[M_i]/N)\, v_i$: the estimator is unbiased for the allocation objective that the *expected* budget split defines, and the second display of (15) is the correct replacement for a "conditional variance" — it is the mean-squared error of $\widehat{V}_{aw}$ around the random target V_M, not a variance around a fixed mean. The rest of Part 1

then goes through verbatim, pathwise. Since $v \mapsto v(1-v)$ is concave on $[0,1]$ and w lies in the probability simplex, Jensen's inequality gives, for every realization,

$$\sum_{i=1}^{k} w_i\, v_i(1-v_i) \;\le\; \left(\sum_{i=1}^{k} w_i v_i\right)\left(1-\sum_{i=1}^{k} w_i v_i\right) = V_M(1-V_M), \tag{16}$$

and a second application of Jensen (now to the concave map $V \mapsto V(1-V)$ under the outer expectation) yields

$$\mathbb{E}\big[(\widehat{V}_{aw}-V_M)^2\big] \;\le\; \frac{\mathbb{E}\big[V_M(1-V_M)\big]}{N} \;\le\; \frac{\mathbb{E}[V_M]\big(1-\mathbb{E}[V_M]\big)}{N} \;\le\; \frac{1}{4N}, \tag{17}$$

with the last inequality strict whenever $\mathbb{E}[V_M] \neq 1/2$. This is Proposition F.1(ii). Qualitatively, concentrated allocation, which pushes the allocated objective V_M toward 0 or 1, lowers the error ceiling below the Bernoulli worst case $1/(4N)$ rather than raising it.

Part 2: rounding mismatch to the real-valued allocation objective. We first record the elementary inequality that drives both this part and the trivial bound (iii).

Lemma F.4 (Zero-sum reweighting)**.** *Let $d \in \mathbb{R}^k$ satisfy $\sum_i d_i = 0$, and let $a \in \mathbb{R}^k$ with $a_{\max} - a_{\min} = \Delta$. Then*

$$\Big|\sum_{i=1}^{k} d_i a_i\Big| \;\le\; \Delta \sum_{i=1}^{k} d_i^{+} \;=\; \frac{\Delta}{2}\,\|d\|_1, \qquad d_i^{+} := \max(d_i, 0). \tag{18}$$

Proof. Because $\sum_i d_i = 0$ we may recentre: $\sum_i d_i a_i = \sum_i d_i(a_i - c)$ for any constant c. Taking $c = (a_{\max} + a_{\min})/2$ gives $|a_i - c| \le \Delta/2$ for all i, whence $|\sum_i d_i a_i| \le \frac{\Delta}{2}\sum_i |d_i|$. The zero-sum constraint also gives $\sum_i d_i^{+} = \sum_i d_i^{-} = \frac{1}{2}\|d\|_1$, which is the stated identity. □

Define the rounding deviations $d_i := \frac{M_i}{N} - q_i$ (we write d_i rather than reusing the symbol δ_i, which Part 3 reserves for centred posterior scores). Largest-remainder rounding of the real-valued targets $\{Nq_i\}$ has more structure than the crude bound $|M_i - Nq_i| \le 1$ records, and that structure is what tightens the constant. Let $f_i := Nq_i - \lfloor Nq_i \rfloor \in [0,1)$ be the fractional parts and $R := N - \sum_i \lfloor Nq_i \rfloor = \sum_i f_i \in \{0, 1, \ldots, k-1\}$ the number of residual units; the rule awards one extra unit to each of the R plans with the largest remainders. Hence $M_i - Nq_i = 1 - f_i > 0$ for the R recipients and $M_i - Nq_i = -f_i \le 0$ for the others, so with $A := \sum_{i\,\text{recipient}} f_i$,

$$\sum_{i=1}^{k} \big(M_i - Nq_i\big)^{+} = R - A. \tag{19}$$

The recipients hold the R largest of the k remainders, so their mean is at least the mean of all k remainders, $A/R \ge \frac{1}{k}\sum_i f_i = R/k$, i.e. $A \ge R^2/k$. Therefore

$$\sum_{i=1}^{k} d_i^{+} \;\le\; \frac{1}{N}\Big(R - \frac{R^2}{k}\Big) = \frac{R(k-R)}{kN} \;\le\; \frac{\lfloor k/2 \rfloor \lceil k/2 \rceil}{kN} \;\le\; \frac{k}{4N}, \tag{20}$$

the middle maximum being attained at $R = \lfloor k/2 \rfloor$. Since $\sum_i d_i = 0$, Theorem F.4 applies with $a = v$ and gives the pathwise rounding bound

$$\left|V_M - \sum_{i=1}^{k} q_i v_i\right| = \left|\sum_{i=1}^{k} d_i v_i\right| \;\le\; \frac{R(k-R)}{kN}\,\Delta_v \;\le\; \frac{k}{4N}\,\Delta_v. \tag{21}$$

This is a factor of four tighter than the bound $\frac{k}{N}\Delta_v$ obtained by summing $|d_i| \le 1/N$ against the full range: one factor of two comes from Theorem F.4 (the deviations sum to zero, so only their positive part, half the total variation, can act), and one from the largest-remainder structure (only R of the k deviations are positive, and their total mass is at most $R(k-R)/k$, never the k units the crude bound allows). At the operating point $k = 4$, $N = 16$ the constant improves from $k/N = 0.25$ to $k/(4N) = 0.0625$ times Δ_v, so at this budget the rounding term is dominated by the softmax term of Part 3 rather than competing with it. Both R and $\{q_i\}$ are $\mathcal{F}_1$-measurable, so (21) holds for every realization of Stage 1.

Remark F.5 (The deployed scheme rounds only the Stage-2 remainder, and its exploration floor is larger than η). Proposition F.1 is stated for the idealized scheme that rounds $\{Nq_i\}$ directly. Algorithm 2 instead fixes the $kM^{(0)}$ Stage-1 rollouts and rounds only $\{N_{\mathrm{rem}}q_i\}$ with $N_{\mathrm{rem}} = N - kM^{(0)}$. Writing $\zeta := kM^{(0)}/N$ for the probe fraction, the deployed target proportions are

$$\bar{q}_i = \frac{M^{(0)}}{N} + \frac{N_{\mathrm{rem}}}{N} q_i = \frac{\zeta}{k} + (1-\zeta)q_i = (1-\zeta)(1-\eta)\,\tilde{q}_i + \frac{\zeta + (1-\zeta)\eta}{k}, \tag{22}$$

which is again a softmax-mixture allocation, with the mixture weight $(1-\eta)$ replaced by $(1-\zeta)(1-\eta)$ and the floor η by $\eta_{\mathrm{eff}} = \zeta + (1-\zeta)\eta \geq \eta$. Two consequences. (a) The rounding bound is unchanged: with $d_i = \frac{M_i}{N} - \bar{q}_i = \frac{1}{N}\big(M_i^{(1)} - N_{\mathrm{rem}}q_i\big)$, the argument leading to (20) applies verbatim to the budget N_{rem} and again yields $\sum_i d_i^+ \leq R(k-R)/(kN) \leq k/(4N)$. (b) Every bias term carrying the prefactor $(1-\eta)$ may be multiplied by $(1-\zeta)$, so the stated bound is conservative for the deployed algorithm: the uniform Stage-1 probe acts as an additional exploration floor. At the operating point ($k = 4$, $M^{(0)} = 2$, $N = 16$, $\eta = 0.05$) this replaces $1-\eta = 0.95$ by $(1-\zeta)(1-\eta) = 0.475$, halving the leading bias term. Positivity is likewise immediate under the deployed scheme: $M_i \geq M^{(0)} \geq 1$, so no plan is ever dropped and $\widehat{V}_i$ is always formed from at least $M^{(0)}$ executions.

Part 3: temperature-aware bias via a softmax expansion. Everything in this part is pathwise in $\mathcal{F}_1$: $\{u_i\}$, $\{q_i\}$ and c are Stage-1 random variables, while $\{v_i\}$, $\bar{v}$, $\widehat{\mathrm{Var}}(v)$ and Δ_v are deterministic. Using $q_i = (1-\eta)\tilde{q}_i + \eta/k$ and $\sum_i (1/k)v_i = \bar{v}$,

$$\sum_{i=1}^{k} q_i v_i - \bar{v} = (1-\eta)\left(\sum_{i=1}^{k} \tilde{q}_i v_i - \bar{v}\right), \tag{23}$$

so it suffices to bound the unregularized softmax bias, and the exploration floor enters only through the multiplicative factor $(1-\eta)$. This identity already gives the trivial bound of Proposition F.1(iii): $\sum_i(\tilde{q}_i - 1/k) = 0$, so Theorem F.4 applies with $d_i = \tilde{q}_i - 1/k$, and $\|d\|_1 \leq 2(1-1/k)$ because the ℓ_1 distance between a probability vector and the uniform vector is at most $2(1-1/k)$ (attained at a point mass); hence $|\sum_i \tilde{q}_i v_i - \bar{v}| \leq \frac{k-1}{k}\Delta_v$ at every temperature, and with (21),

$$\big|V_M - \bar{v}\big| \;\leq\; (1-\eta)\frac{k-1}{k}\Delta_v + \frac{k}{4N}\Delta_v \qquad \text{almost surely.} \tag{24}$$

For the refined statement, let $\delta_i := u_i - \bar{u}$ with $\bar{u} = \frac{1}{k}\sum_i u_i$, so $\sum_i \delta_i = 0$, and write $\tau := \tau_{\mathrm{alloc}}$ and $\tilde{q}_i = e^{u_i/\tau}/\sum_\ell e^{u_\ell/\tau} = e^{\delta_i/\tau}/\sum_\ell e^{\delta_\ell/\tau}$. Under the high-temperature condition $c := \max_i |\delta_i|/\tau < 1$, expanding the exponentials,

$$e^{\delta_i/\tau} = 1 + \frac{\delta_i}{\tau} + \frac{\delta_i^2}{2\tau^2} + \mathcal{O}(c^3), \tag{25}$$

$$\sum_{\ell=1}^{k} e^{\delta_\ell/\tau} = k + \frac{1}{\tau}\underbrace{\sum_{\ell}\delta_\ell}_{=0} + \frac{S_2}{2\tau^2} + \mathcal{O}(kc^3) = k + \frac{S_2}{2\tau^2} + \mathcal{O}(kc^3), \tag{26}$$

where $S_2 := \sum_\ell \delta_\ell^2 = k\,\widehat{\mathrm{Var}}(u)$. A first-order Neumann expansion of the reciprocal, $\big(k + S_2/(2\tau^2) + \cdots\big)^{-1} = \frac{1}{k}\big(1 - \frac{S_2}{2k\tau^2} + \mathcal{O}(c^3)\big)$, multiplied against the numerator and discarding the third-order cross term $\delta_i S_2/(2k\tau^3) = \mathcal{O}(c^3/k)$, gives

$$\tilde{q}_i = \frac{1}{k}\left(1 + \frac{\delta_i}{\tau} + \frac{\delta_i^2 - S_2/k}{2\tau^2}\right) + \mathcal{O}(c^3/k), \tag{27}$$

and since $\sum_i(\tilde{q}_i - 1/k) = 0$ we may recentre v by $\bar{v}$ before summing,

$$\begin{aligned}
\sum_{i=1}^{k} \tilde{q}_i v_i - \bar{v} &= \sum_{i=1}^{k}\big(\tilde{q}_i - \tfrac{1}{k}\big)\,(v_i - \bar{v}) \\
&= \frac{1}{k\tau}\sum_{i=1}^{k}\delta_i(v_i - \bar{v}) + \mathcal{O}\big(c^2(v_{\max} - v_{\min})\big) \\
&= \frac{\widehat{\mathrm{Cov}}(u,v)}{\tau} + \mathcal{O}\big(c^2(v_{\max} - v_{\min})\big),
\end{aligned} \tag{28}$$

where $\widehat{\mathrm{Cov}}(u,v) := \frac{1}{k}\sum_i (u_i-\bar{u})(v_i-\bar{v})$ is the empirical covariance between posterior score and true plan value. The second-order term in (28) admits an explicit constant: the weights $\delta_i^2 - S_2/k$ also sum to zero, so Theorem F.4 gives

$$\left|\frac{1}{2k\tau^2}\sum_{i=1}^{k}\big(\delta_i^2-\tfrac{S_2}{k}\big)(v_i-\bar{v})\right| \;\le\; \frac{\Delta_v}{2}\cdot\frac{1}{2k\tau^2}\sum_{i=1}^{k}\big|\delta_i^2-\tfrac{S_2}{k}\big| \;\le\; \frac{\Delta_v}{2}\cdot\frac{2S_2}{2k\tau^2} = \frac{\widehat{\mathrm{Var}}(u)}{2\tau^2}\Delta_v \;\le\; \frac{c^2}{2}\Delta_v, \tag{29}$$

using $\sum_i |\delta_i^2 - S_2/k| \le 2S_2$ and $\widehat{\mathrm{Var}}(u) = \frac{1}{k}\sum_i \delta_i^2 \le \max_i \delta_i^2 = (c\tau)^2$. The $\mathcal{O}(c^2\Delta_v)$ term in (28) is therefore at most $\frac{c^2}{2}\Delta_v + \mathcal{O}(c^3\Delta_v)$.

Remark F.6 (A non-asymptotic envelope for the discarded terms). The expansion above is the sharp description of the small-c regime, but it is worth recording that the discarded terms cannot hide a diverging constant. Without any expansion, $|\delta_i|/\tau \le c$ implies $e^{\delta_i/\tau} \in [e^{-c}, e^{c}]$ and hence $k\tilde{q}_i \in [e^{-2c}, e^{2c}]$, so $|\tilde{q}_i - 1/k| \le (e^{2c}-1)/k$ and, by Theorem F.4, $\big|\sum_i \tilde{q}_i v_i - \bar{v}\big| \le \frac{\Delta_v}{2}\big(e^{2c}-1\big)$ for every c. This envelope is exact and always valid; it is linear rather than quadratic in c for small c, so it certifies rather than replaces (28). Taken with (24), it shows the bias is bounded by a finite, explicit quantity under every parameter setting, and that the refined bound is the binding one precisely in the small-spread, high-temperature regime the expansion describes.

Part 4: decomposition of $\widehat{\mathrm{Cov}}(u,v)$ via calibration error. Write $u_i = \mu_i + \lambda\sigma_i = v_i + \varepsilon_i + \lambda\sigma_i$ with calibration error $\varepsilon_i := \mu_i - v_i$, so that, by bilinearity of the empirical covariance,

$$\widehat{\mathrm{Cov}}(u,v) = \widehat{\mathrm{Var}}(v) + \widehat{\mathrm{Cov}}(\varepsilon, v) + \lambda\,\widehat{\mathrm{Cov}}(\sigma, v). \tag{30}$$

The first term is the signal: it is exactly the plan-value dispersion the allocation is meant to exploit, and it is deterministic. The other two are errors. By Cauchy–Schwarz, and using that an empirical variance is bounded by the corresponding second moment,

$$|\widehat{\mathrm{Cov}}(\varepsilon, v)| \le \sqrt{\widehat{\mathrm{Var}}(\varepsilon)\,\widehat{\mathrm{Var}}(v)} \le \varepsilon_{\max}\sqrt{\widehat{\mathrm{Var}}(v)}, \tag{31}$$

$$|\widehat{\mathrm{Cov}}(\sigma, v)| \le \sqrt{\widehat{\mathrm{Var}}(\sigma)\,\widehat{\mathrm{Var}}(v)}. \tag{32}$$

For a Beta(α,β) posterior with $\alpha+\beta = M^{(0)}+2$ (uniform prior on top of $M^{(0)}$ Stage-1 probes), $\sigma_i^2 = \mu_i(1-\mu_i)/(M^{(0)}+3) \le 1/(4(M^{(0)}+3))$, so $\widehat{\mathrm{Var}}(\sigma) \le \max_i \sigma_i^2 \le 1/(4(M^{(0)}+3))$ and $\sqrt{\widehat{\mathrm{Var}}(\sigma)} \le 1/(2\sqrt{M^{(0)}+3})$ — a bound that holds surely, not merely in expectation, because it uses only $\mu_i \in [0,1]$. Combining these with Eqs. (23)–(28) yields the pathwise softmax bias bound

$$\left|\sum_{i=1}^{k} q_i v_i - \bar{v}\right| \le \frac{1-\eta}{\tau_{\mathrm{alloc}}}\left(\widehat{\mathrm{Var}}(v) + \Big(\varepsilon_{\max} + \frac{\lambda}{2\sqrt{M^{(0)}+3}}\Big)\sqrt{\widehat{\mathrm{Var}}(v)}\right) + (1-\eta)\Big(\frac{c^2}{2}\Delta_v + \mathcal{O}\big(c^3\Delta_v\big)\Big), \tag{33}$$

valid for every realization of $\mathcal{F}_1$ on which $c < 1$. The only random quantities remaining on the right-hand side are $\varepsilon_{\max}$ and c, both $\mathcal{F}_1$-measurable and both entering monotonically.

Lemma F.7 (Explicit control of the calibration term)**.** *With $\alpha_0 = \beta_0 = 1$ and $M^{(0)}$ Bernoulli probes per plan,*

$$\mathbb{E}[\varepsilon_{\max}] = \mathbb{E}\Big[\max_i |\mu_i - v_i|\Big] \;\le\; \sqrt{\frac{\log(2k)}{2M^{(0)}}} + \frac{1}{M^{(0)}+2}. \tag{34}$$

Proof. Let $\hat{p}_i := s_i/M^{(0)}$ be the probe frequency. Since $\mu_i = (1+s_i)/(M^{(0)}+2)$, we have $\mu_i - \hat{p}_i = \big(M^{(0)} - 2s_i\big)/\big(M^{(0)}(M^{(0)}+2)\big)$ and $|M^{(0)} - 2s_i| \le M^{(0)}$, so the prior-shrinkage term is at most $1/(M^{(0)}+2)$ deterministically. Each $\hat{p}_i - v_i$ is an average of $M^{(0)}$ independent centred variables supported on an interval of length one, hence sub-Gaussian with variance proxy $1/(4M^{(0)})$ by Hoeffding's lemma, and the standard maximal inequality applied to the $2k$ variables $\pm(\hat{p}_i - v_i)$ gives $\mathbb{E}\max_i |\hat{p}_i - v_i| \le \sqrt{\log(2k)/(2M^{(0)})}$. The triangle inequality for the maximum completes the proof. □

Remark F.8 (At the deployed probe size the bias is controlled by construction, not by posterior concentration). Lemma F.7 is informative only for reasonably large probes: at the deployed setting $k = 4$, $M^{(0)} = 2$ it evaluates to roughly 0.97, which is no better than the trivial $\varepsilon_{\max} \le 1$. This is a statement about what a two-sample posterior can guarantee, not a defect of the estimator. At the operating point the bias is held in check by the exploration floor rather than by posterior accuracy: the prefactor $(1-\zeta)(1-\eta) = 0.475$ of Theorem F.5, together with the trivial bound (24), caps the bias however badly calibrated the posterior is, while (13) is the binding bound in the small-spread regime, where it shows the bias vanishing quadratically as the plan values equalise, exactly the regime in which a biased allocation would otherwise be hardest to justify. The PVE warm-up (Algorithm 1) is what makes the posterior means informative in practice; the theory here does not assume it.

Part 5: assembling the parts under the law of total expectation. The quantity of interest, $\mathbb{E}[\widehat{V}_{aw}] - \bar{v}$, is deterministic, whereas Parts 2–4 bound random quantities; the two are joined by taking expectations, which is legitimate precisely because Parts 2–4 are pathwise. By Theorem F.3, $\mathbb{E}[\widehat{V}_{aw}] = \mathbb{E}[V_M]$, so by Jensen's inequality for the absolute value (equivalently, the triangle inequality for expectations),

$$\big|\mathbb{E}[\widehat{V}_{aw}] - \bar{v}\big| = \big|\mathbb{E}\big[V_M - \bar{v}\big]\big| \;\le\; \mathbb{E}\big|V_M - \bar{v}\big| \;\le\; \mathbb{E}\left[\left|V_M - \sum_i q_i v_i\right|\right] + \mathbb{E}\left[\left|\sum_i q_i v_i - \bar{v}\right|\right], \tag{35}$$

where the inner triangle inequality is applied realization by realization. Substituting the pathwise rounding bound (21) into the first expectation and the pathwise softmax bias bound (33) into the second, and using that $\widehat{\mathrm{Var}}(v)$, Δ_v and $\lambda/(2\sqrt{M^{(0)}+3})$ are deterministic while $\varepsilon_{\max}$ and c are not, gives

$$\begin{aligned}\big|\mathbb{E}[\widehat{V}_{aw}] - \bar{v}\big| \;\le\; & \frac{1-\eta}{\tau_{\mathrm{alloc}}}\left(\widehat{\mathrm{Var}}(v) + \left(\mathbb{E}[\varepsilon_{\max}] + \frac{\lambda}{2\sqrt{M^{(0)}+3}}\right)\sqrt{\widehat{\mathrm{Var}}(v)}\right) \\ & + \frac{1-\eta}{2}\,\mathbb{E}\big[c^2\big]\Delta_v + \mathcal{O}\big(\mathbb{E}[c^3]\,\Delta_v\big) + \mathbb{E}\left[\frac{R(k-R)}{kN}\right]\Delta_v,\end{aligned} \tag{36}$$

which is (13) once the last expectation is bounded by its worst case $k/(4N)$ and the two c-terms are collected into $\mathcal{O}(\mathbb{E}[c^2]\Delta_v)$. The same substitution with (24) in place of (33) gives Proposition F.1(iii), which requires no regime assumption. This completes the proof.

Three features of the assembled bound are worth stating plainly, because they are exactly the places where a conditional argument is silent. (a) Nothing was assumed about the dependence between $\{M_i\}$ and $\{v_i\}$, or between $\{M_i\}$ and the Stage-1 rewards; that dependence is real, is what makes the allocation adaptive, and is handled by the predictability of a_t rather than assumed away. (b) The bias is measured against $\bar{v}$, the uniform-plan objective, so it is the price of allocation, not an estimation error — the estimation error is (17), and it is lower under concentrated allocation. (c) All $\mathcal{F}_1$-randomness surviving into (13) is confined to $\mathbb{E}[\varepsilon_{\max}]$ and $\mathbb{E}[c^2]$, both moments of Stage-1 quantities that shrink as $M^{(0)}$ grows and as τ_{alloc} is raised.

Corollary F.9 (Mean-squared error around the uniform-plan objective)**.** *Under the assumptions of Proposition F.1, let B denote any almost-sure pathwise bound on $|V_M - \bar{v}|$ (for instance the right-hand side of* (24)*, or* (33) *plus* (21) *on the high-temperature event). Then*

$$\Big(\mathbb{E}\big[(\widehat{V}_{aw} - \bar{v})^2\big]\Big)^{1/2} \;\le\; \Big(\mathbb{E}\big[(\widehat{V}_{aw} - V_M)^2\big]\Big)^{1/2} + \big\|V_M - \bar{v}\big\|_2 \;\le\; \frac{1}{2\sqrt{N}} + B. \tag{37}$$

Proof. Minkowski's inequality in L^2 applied to $\widehat{V}_{aw} - \bar{v} = (\widehat{V}_{aw} - V_M) + (V_M - \bar{v})$, followed by (17) for the first term and $\|V_M - \bar{v}\|_2 \le B$ for the second. The cross term is not zero in general ($V_M - \bar{v}$ is $\mathcal{F}_1$-measurable while $\widehat{V}_{aw} - V_M$ is not centred given $\mathcal{F}_1$, since the Stage-1 rewards enter both), so the familiar bias–variance decomposition is unavailable and the L^2 triangle inequality is used instead. □

Remark F.10 (When the high-temperature regime is not a sure event). Proposition F.1(iv) assumes $c < 1$ almost surely. A sufficient deterministic condition is $\tau_{\mathrm{alloc}} > 1 + \lambda/(2\sqrt{M^{(0)}+3})$, since $u_i \in [0,\, 1 + \lambda/(2\sqrt{M^{(0)}+3})]$ surely and $\max_i |u_i - \bar{u}| \le u_{\max} - u_{\min}$. At the deployed setting

($\lambda = 1.0$, $M^{(0)} = 2$, $\tau_{\text{alloc}} = 0.7$) this worst-case condition would require $\tau_{\text{alloc}} \gtrsim 1.22$ and is therefore not met: the regime is an event $\mathcal{E} := \{c < 1\}$, determined by the realized posterior-score spread rather than by its worst case. The honest unconditional statement is then the mixture

$$\left|\mathbb{E}[\widehat{V}_{aw}] - \bar{v}\right| \;\le\; \mathbb{E}\big[\mathbf{1}_{\mathcal{E}}\, B_{\text{ref}}\big] + \Pr(\mathcal{E}^c)\,(1-\eta)\frac{k-1}{k}\Delta_v + \frac{k}{4N}\Delta_v, \tag{38}$$

with B_{ref} the softmax term of (33): the refined bound applies where the expansion is valid and the trivial bound elsewhere, the two being combined by the same law-of-total-expectation step as in Part 5. On $\mathcal{E}^c$ the exact envelope of Theorem F.6 applies in any case, so the bias stays finite and explicitly bounded under every parameter setting.

G The One-Way ANOVA Variance Identity

Theorem 4.1(a) is stated over a single algebraic identity: the variance of a flat, grand-mean-centred advantage on a batch organised as k plans of M rollouts. This appendix derives that identity in full, derives the two instantiations of it that the main text labels, shows that both share the same asymptotics, and records the exact covariance structure that Appendix H needs. The identity is the only place in the theory where a distributional assumption could have entered and does not: everything below uses second moments only, so Gaussian and Bernoulli rewards are covered by the same statement.

G.1 Model and estimators

Fix a problem x and a batch of N execution rollouts organised as k latent plans with $M = N/k$ rollouts each (balanced allocation; this is the PVE regime, $\omega_i \equiv 1$). Write

$$r_{i,j} \;=\; v_i + \varepsilon_{i,j}, \qquad i \in [k],\; j \in [M], \tag{39}$$

where v_i is the *plan-conditional reward mean* of plan i and $\varepsilon_{i,j}$ is the within-plan residual. We use v_i rather than the customary $\mu + \alpha_i$ so that no symbol collides with the policy parameter θ. The model is specified by moments alone:

$$\mathbb{E}[v_i] = \mu, \quad \sigma_B^2 := \mathrm{Var}(v_i), \quad \sigma_W^2 := \mathbb{E}\big[\mathrm{Var}(r \mid v_i)\big], \quad \sigma^2 := \sigma_B^2 + \sigma_W^2, \quad \rho_r := \frac{\sigma_B^2}{\sigma^2}, \tag{40}$$

with $\{v_i\}$ i.i.d. across plans, $\{\varepsilon_{i,j}\}$ mean-zero and uncorrelated across all pairs $(i,j) \neq (i',j')$, and $\varepsilon \perp v$. The decomposition $\sigma^2 = \sigma_B^2 + \sigma_W^2$ is the law of total variance applied to r, so it holds without any parametric assumption (Scheffé, 1959; Searle et al., 2006). For binary execution rewards the two components specialise to $\sigma_B^2 = \mathrm{Var}(v_i)$ and $\sigma_W^2 = \mathbb{E}[v_i(1-v_i)]$, with $\sigma^2 = \bar{v}(1-\bar{v})$ where $\bar{v} = \mathbb{E}[v_i]$; nothing below changes.

The three estimators compared in the paper are

$$\bar{r} = \frac{1}{N}\sum_{i,j} r_{i,j}, \qquad \widehat{V}_i = \frac{1}{M}\sum_{m=1}^{M} r_{i,m}, \qquad \bar{V} = \frac{1}{k}\sum_{i=1}^{k} \widehat{V}_i \;=\; \bar{r} \;\text{(under balance)}, \tag{41}$$

giving the flat advantage, the code-level (plan-conditioned) advantage and the plan-level advantage

$$\widehat{A}^{\text{flat}}_{i,j} = r_{i,j} - \bar{r}, \qquad \widehat{A}_{\text{code},i,j} = r_{i,j} - \widehat{V}_i, \qquad \widehat{A}_{\text{plan},i} = \widehat{V}_i - \bar{V}. \tag{42}$$

G.2 The identity

Lemma G.1 (One-way ANOVA variance identity)**.** *Under Equation* (39)*–Equation* (40) *with balanced $M = N/k$,*

$$\mathrm{Var}\big(\widehat{A}^{\text{flat}}_{i,j}\big) = \frac{k-1}{k}\,\sigma_B^2 + \frac{N-1}{N}\,\sigma_W^2, \tag{43}$$

$$\mathrm{Var}\big(\widehat{A}_{\text{code},i,j}\big) = \frac{M-1}{M}\,\sigma_W^2 \;=\; \frac{M-1}{M}\,(1-\rho_r)\,\sigma^2, \tag{44}$$

$$\mathrm{Var}\big(\widehat{A}_{\text{plan},i}\big) = \frac{k-1}{k}\Big(\sigma_B^2 + \frac{\sigma_W^2}{M}\Big). \tag{45}$$

Proof. Write $\bar{v} = \frac{1}{k}\sum_i v_i$, $\bar{\varepsilon} = \frac{1}{N}\sum_{i,j}\varepsilon_{i,j}$ and $\bar{\varepsilon}_{i\cdot} = \frac{1}{M}\sum_m \varepsilon_{i,m}$. Under balance $\bar{r} = \bar{v} + \bar{\varepsilon}$, so

$$\widehat{A}^{\text{flat}}_{i,j} = (v_i - \bar{v}) + (\varepsilon_{i,j} - \bar{\varepsilon}), \tag{46}$$

and the two brackets are uncorrelated because $\varepsilon \perp v$. For the first,

$$\mathrm{Var}(v_i - \bar{v}) = \mathrm{Var}\Big(\tfrac{k-1}{k}v_i - \tfrac{1}{k}\textstyle\sum_{i'\neq i} v_{i'}\Big) = \Big(\tfrac{k-1}{k}\Big)^2\sigma_B^2 + \tfrac{k-1}{k^2}\sigma_B^2 = \tfrac{(k-1)k}{k^2}\sigma_B^2 = \tfrac{k-1}{k}\sigma_B^2,$$

and the identical computation over N uncorrelated residuals gives $\mathrm{Var}(\varepsilon_{i,j} - \bar{\varepsilon}) = \frac{N-1}{N}\sigma_W^2$. Summing yields Equation (43).

For Equation (44), the plan effect cancels exactly rather than approximately: $\widehat{A}_{\text{code},i,j} = v_i + \varepsilon_{i,j} - v_i - \bar{\varepsilon}_{i\cdot} = \varepsilon_{i,j} - \bar{\varepsilon}_{i\cdot}$, which involves no v at all, and the same centring computation over M residuals gives $\frac{M-1}{M}\sigma_W^2$. Substituting $\sigma_W^2 = (1-\rho_r)\sigma^2$ gives the stated form. The cancellation is exact rather than asymptotic, and it is the mechanism the whole method rests on.

For Equation (45), $\widehat{V}_i = v_i + \bar{\varepsilon}_{i\cdot}$ are i.i.d. across i with variance $\sigma_B^2 + \sigma_W^2/M$, and $\widehat{A}_{\text{plan},i}$ centres k such variables, contributing the same $\frac{k-1}{k}$ factor. □

The second moment of the plan-level advantage carries an estimation-noise term σ_W^2/M on top of the between-plan variance σ_B^2: estimating each plan value from M samples is not free, and this term is what Appendix H must carry into the PROSE-side gradient bound rather than discard.

Lemma G.2 (Exact covariance structure)**.** *Under the same conditions, for* $(i,j) \neq (i',j')$*,*

$$\mathrm{Cov}\big(\widehat{A}^{\text{flat}}_{i,j}, \widehat{A}^{\text{flat}}_{i',j'}\big) = \begin{cases} \dfrac{k-1}{k}\sigma_B^2 - \dfrac{\sigma_W^2}{N}, & i = i',\ j \neq j', \\ -\dfrac{\sigma_B^2}{k} - \dfrac{\sigma_W^2}{N}, & i \neq i', \end{cases} \tag{47}$$

$$\mathrm{Cov}\big(\widehat{A}_{\text{code},i,j}, \widehat{A}_{\text{code},i',j'}\big) = \begin{cases} -\sigma_W^2/M, & i = i',\ j \neq j', \\ 0, & i \neq i', \end{cases} \tag{48}$$

$$\mathrm{Cov}\big(\widehat{A}_{\text{plan},i}, \widehat{A}_{\text{plan},i'}\big) = -\frac{\sigma_B^2 + \sigma_W^2/M}{k} \qquad (i \neq i').$$

Each family sums to zero over its index set, i.e. $\sum_{i',j'}\mathrm{Cov}(\widehat{A}^{\text{flat}}_{i,j}, \widehat{A}^{\text{flat}}_{i',j'}) = 0$ *and likewise within each plan for* $\widehat{A}_{\text{code}}$ *and across plans for* $\widehat{A}_{\text{plan}}$*.*

Proof. All entries follow from Equation (46) and $\mathrm{Cov}(v_i, \bar{v}) = \sigma_B^2/k$, $\mathrm{Cov}(\varepsilon_{i,j}, \bar{\varepsilon}) = \sigma_W^2/N$. For the same-plan entry, $\mathrm{Cov}(v_i - \bar{v}, v_i - \bar{v}) = \frac{k-1}{k}\sigma_B^2$ and $\mathrm{Cov}(\varepsilon_{i,j} - \bar{\varepsilon}, \varepsilon_{i,j'} - \bar{\varepsilon}) = 0 - \frac{\sigma_W^2}{N} - \frac{\sigma_W^2}{N} + \frac{\sigma_W^2}{N} = -\frac{\sigma_W^2}{N}$; for the cross-plan entry, $\mathrm{Cov}(v_i - \bar{v}, v_{i'} - \bar{v}) = -\sigma_B^2/k$ by the same three-term expansion. The $\widehat{A}_{\text{code}}$ entries follow from $\widehat{A}_{\text{code},i,j} = \varepsilon_{i,j} - \bar{\varepsilon}_{i\cdot}$, which is a function of plan i's residuals only, hence exactly zero cross-plan covariance with no $O(1/N)$ remainder. For the zero-sum check on the flat family, collect the σ_B^2 coefficients over the one diagonal, $M-1$ same-plan and $N-M$ cross-plan terms: $\frac{k-1}{k} + (M-1)\frac{k-1}{k} - (N-M)\frac{1}{k} = \frac{M(k-1)-(N-M)}{k} = \frac{N-N}{k} = 0$, and the σ_W^2 coefficients: $\frac{N-1}{N} - \frac{M-1}{N} - \frac{N-M}{N} = 0$. □

Lemma G.2 is stated exactly, with no $O(1/N)$ terms dropped, because Appendix H needs the zero-sum property: a truncation that discards the cross-plan entries breaks $\sum_{i,j}\widehat{A}^{\text{flat}}_{i,j} = 0$ and thereby overstates the GRPO-side variance. We return to that point in Remark H.4.

G.3 Two instantiations of the flat comparator

Equation (43) is a statement about a batch organised as k plans of M rollouts. Which (k, M) one substitutes determines which question is being answered, and the two substitutions the paper uses are labelled throughout.

(i) Same-batch comparator (k plans of size M). Both estimators are recomputed on one stored clustered batch. This is the quantity a recomputation protocol actually estimates, because the logged rollouts already carry their plan assignments. Dividing Equation (44) by Equation (43),

$$R_{\text{same}} = \frac{\frac{M-1}{M}(1-\rho_r)}{\frac{k-1}{k}\rho_r + \frac{N-1}{N}(1-\rho_r)}. \tag{49}$$

At the training operating point $k = M = 4$, $N = 16$, $\rho_r = 0.72$ the denominator is $\frac{3}{4}(0.72) + \frac{15}{16}(0.28) = 0.54 + 0.2625 = 0.8025$ and the numerator is $\frac{3}{4}(0.28) = 0.2100$, so $R_{\text{same}} = 0.2100/0.8025 = 0.262$ and $1 - R_{\text{same}} = 0.738$.

(ii) GRPO-as-deployed comparator ($k = N$, $M = 1$). GRPO as actually run draws N i.i.d. rollouts with no plan conditioning; in the identity this is $k = N$, $M = 1$, i.e. every rollout is its own latent plan. Substituting into Equation (43) the two terms merge exactly:

$$\frac{k-1}{k}\sigma_B^2 + \frac{N-1}{N}\sigma_W^2 \Big|_{k=N} = \frac{N-1}{N}\big(\sigma_B^2 + \sigma_W^2\big) = \frac{N-1}{N}\,\sigma^2. \tag{50}$$

This exact reduction is the reason the two figures quoted in the paper are both correct rather than in conflict: the deployed comparator is the same identity evaluated at a different (k, M). Comparing PROSE's clustered $\frac{M-1}{M}(1-\rho_r)\sigma^2$ at $M = 4$ against it gives

$$R_{\text{dep}} = \frac{\frac{M-1}{M}(1-\rho_r)\sigma^2}{\frac{N-1}{N}\sigma^2} = \frac{(M-1)(1-\rho_r)\,N}{M(N-1)} = \frac{3(0.28)(16)}{4(15)} = 0.224, \qquad 1 - R_{\text{dep}} = 0.776. \tag{51}$$

Because the numerator carries $M = 4$ and the denominator $M = 1$, this is a *cross-design* quantity: it answers what the hierarchical advantage buys over the flat advantage as GRPO samples it, and it is not the number a same-batch recomputation returns. We label it as such wherever it appears; Appendix I.1 discusses the choice between the two.

Table 9: The two instantiations of Equation (43), evaluated at $\rho_r = 0.72$, $N = 16$. Both rows use the same identity; they differ only in the (k, M) at which the flat comparator is evaluated. The code-level row is always PROSE's clustered $M = 4$ estimator, so row (ii) compares across designs. The rightmost column is the common fixed-M limit as the number of plans grows with $M = 4$ held fixed. $V_{\text{flat}} := \text{Var}(\widehat{A}^{\text{flat}}_{i,j})/\sigma^2$ and $V_{\text{code}} := \text{Var}(\widehat{A}_{\text{code},i,j})/\sigma^2$.

Flat comparator	k	M	V_{flat}	V_{code}	R	$1-R$	$\lim(1-R)$
(i) same-batch (clustered)	4	4	0.8025	0.2100	0.262	0.738	0.79
(ii) GRPO-as-deployed	16	1	0.9375	0.2100	0.224	0.776	0.79

G.4 Shared asymptotics

The two instantiations differ at finite batch size and agree in the limit. Hold M fixed and let the number of plans grow ($k \to \infty$, hence $N \to \infty$). In Equation (49) the factors $\frac{k-1}{k}$ and $\frac{N-1}{N}$ both tend to 1, so the denominator tends to $\rho_r + (1-\rho_r) = 1$ and

$$1 - R_{\text{same}} \;\longrightarrow\; 1 - \frac{(M-1)(1-\rho_r)}{M} \;=\; \frac{M - (M-1) + (M-1)\rho_r}{M} \;=\; \frac{1 + (M-1)\rho_r}{M}, \tag{52}$$

and in Equation (51) the factor $N/(N-1) \to 1$ gives the identical limit. The two comparators therefore carry the same asymptotic content and differ only in the finite-batch correction terms; at $M = 4$, $\rho_r = 0.72$ the shared limit is $[1 + 3(0.72)]/4 = 3.16/4 = 0.79$.

Remark G.3 (The fixed-M limit argues against us, and we report it). The clean headline $1 - R \to \rho_r$ requires $M \to \infty$ as well as $k \to \infty$: from Equation (52),

$$\frac{1 + (M-1)\rho_r}{M} - \rho_r = \frac{1-\rho_r}{M} \;\geq\; 0,$$

with equality only at $\rho_r = 1$. So at any finite plan size the fixed-M reduction fraction overstates the asymptotic ρ_r rather than understating it, and at our operating point it does so by $(1-0.72)/4 = 0.07$ (0.79 versus 0.72). We report the fixed-M limit rather than the more flattering asymptotic value. The finite-batch figures in Table 9 (0.738 and 0.776) sit below the fixed-M limit in turn, because the self-centred baselines cost the estimator one degree of freedom at each level.

G.5 Protocol matching

The measured advantage-variance ratio reported in Section 5 was obtained by recomputing both estimators on the stored rollouts of the $k{=}4$, $M{=}4$, $N{=}16$ batch — the training operating point, and the same batch used for the score-correlation proxies of Appendix L. The protocol-matched prediction is therefore the same-batch instantiation, $R_{\text{same}} = 0.262$, against a measured 0.26. The comparison has no free parameter: ρ_r enters from the independent ICC measurement of Appendix K and (k, M, N) from the training configuration.

Two honesty notes. First, 0.26 carries two significant figures and no reported standard error, so the agreement should be read as an internal-consistency check on the random-effects specification, not as a precision claim. Second, ρ_r and the recomputed ratio come from the same pool of problems, so the check is in-sample in its parameter; a split-half version (ρ_r estimated on a random half of the held-out problems and used to predict the ratio on the disjoint half) is specified as a safeguard in Appendix K.4 but has not been run, and nothing above should be read as reporting it. The deployed instantiation $R_{\text{dep}} = 0.224$ is reported separately and labelled as the cross-design quantity it is, so that the two are never paired against a single measurement.

G.6 Relation to the optimality statement

Equation (44) gives the second moment attained by PROSE's code-level advantage. Proposition I.1 (Appendix I) shows that $\frac{M-1}{M}\sigma_W^2$ is the minimum-variance value *within the symmetric subtractive class* — advantages of the form $r_{i,j} - b(\{r_{i,m}\}_{m\le M})$ with b permutation-invariant within a plan and plan-conditionally unbiased for v_i. The qualifier is load-bearing: process-level or partial-credit rewards (token-level process reward models, test-case-level partial credit) supply information that leaves this class entirely (Ye et al., 2025), and the optimality claim does not extend to them.

H Gradient-Level Extension: Proof of Theorem 4.1(c)

Theorem 4.1(a)–(b) are statements about the variance of a scalar advantage. This appendix carries them to the quantity that actually drives learning, the policy-gradient estimator, and proves part (c). Two things are new relative to a textbook variance comparison. First, rollouts that share a plan emit the same plan-prefix tokens and therefore inherit a common score component, so the "orthogonal scores" simplification (Greensmith et al., 2004) is not available and the within-plan score correlation must be carried through. Second, the step that compares the two sides requires a condition on the relative size of the plan-prefix and code-suffix score norms. That condition was previously implicit in the algebra; it is stated here as **(A3)**, proved where it is used, and checked against the measured proxies.

H.1 Setup, conventions, and assumptions

Let $J(\theta) = \mathbb{E}_{x,y}[r(y)]$ be the population objective under π_θ, assumed L-smooth in θ, let $g := \nabla_\theta J(\theta)$, and let $\hat{g}$ be an unbiased estimator of g of the standard policy-gradient form

$$\hat{g} = \frac{1}{N}\sum_{i,j} h_{i,j}\, A_{i,j}, \qquad \Sigma := \mathbb{E}\big[\|\hat{g} - g\|^2\big], \tag{53}$$

where $A_{i,j}$ is the advantage attached to rollout (i, j) and $h_{i,j}$ its score. Rewards follow the one-way random-effects model of Appendix G.1, and both estimators use the same balanced allocation $M_i = M = N/k$, so the comparison isolates the advantage form rather than the budget.

Convention (C1): expected quantities throughout. Every score norm and inner product below denotes an *expectation* under within-plan sampling $p_i \sim \pi_\theta(\cdot \mid x)$, $y_{i,j} \sim \pi_\theta(\cdot \mid x, p_i)$:

$$\|h^\bullet\|^2 := \mathbb{E}\big\|h^\bullet_{i,j}\big\|^2, \qquad \big\langle h^\bullet_{i,j}, h^\bullet_{i,j'}\big\rangle := \mathbb{E}\big\langle h^\bullet_{i,j}, h^\bullet_{i,j'}\big\rangle, \qquad \bullet \in \{\text{flat}, \text{cond}, \text{plan}\}, \tag{54}$$

together with the within-method uniformity $\mathbb{E}\|h^{\text{flat}}_{i,j}\|^2 = \|h^{\text{flat}}\|^2$ and $\mathbb{E}\|h^{\text{cond}}_{i,j}\|^2 = \|h^{\text{cond}}\|^2$ for all (i, j). Both sides of every display below are then deterministic, which matters: Σ is itself an expectation, and bounding it by a realised random norm would compare objects of different types.

The earlier presentation left the norms as realised vectors at a linearisation point and repaired the mismatch afterwards with a concentration argument; under Equation (54) no such repair is needed, and the structural identities of Remark H.2 hold exactly rather than up to $O(1/\sqrt{M})$.

Convention (C2): score–advantage decoupling. In forming Σ we treat scores and advantages as conditionally uncorrelated given the plan assignment, so that $\mathbb{E}[\langle h_{i,j}, h_{i',j'}\rangle A_{i,j}A_{i',j'}] = \mathbb{E}\langle h_{i,j}, h_{i',j'}\rangle \cdot \mathrm{Cov}(A_{i,j}, A_{i',j'})$. This is the standard simplification in variance analyses of policy-gradient estimators (Greensmith et al., 2004); it is an approximation, not an identity, and it is applied identically to both estimators so that the residual error is common-mode.

Assumption H.1 (Gradient-level structural assumptions)**.**

- **(A1)** Cross-plan score orthogonality: $\mathbb{E}\langle h^{\bullet}_{i,j}, h^{\bullet}_{i',j'}\rangle = 0$ for $i \neq i'$. Exact under the idealised model in which distinct plans share no generated prefix; only approximate in practice, because all rollouts share the problem description. Remark H.3 derives the sign and magnitude of the resulting correction.
- **(A2)** Non-negative within-plan score correlation: $\rho_h^{\mathrm{flat}} \geq 0$.
- **(A3)** Score-norm condition: $\|h^{\mathrm{cond}}\|^2 \geq (M-1)\big(1-\rho_h^{\mathrm{flat}}\big)\|h^{\mathrm{plan}}\|^2$. This is a *sufficient* form; the exact requirement carries an additional factor $1+(M-1)\rho_h^{\mathrm{flat}}$ on the left (Equation (69)).

Theorem 4.1(c), restated. Under (C1)–(C2) and (A1)–(A3), with step sizes optimised separately for each estimator, the per-step policy-improvement ratio satisfies $\Delta J^{\mathrm{PROSE}}/\Delta J^{\mathrm{GRPO}} \geq 1$, with GRPO's gradient variance inflated by the gradient design effect $D_{\mathrm{eff}} = 1 + (M-1)\rho_h^{\mathrm{flat}}\rho_r$ relative to what the same flat scores would give on a batch carrying no within-plan reward or score correlation, against the counterfactually clustered comparator. The inequality is strict when $\rho_r > 0$ and (A3) holds strictly.

(A3) is stated because it is the one gradient-level condition that is checkable at current model scales; Appendix H.7 evaluates it, under two readings of the score-norm ratio that our proxies do not reconcile. It was previously left implicit inside the proof, which was a defect of presentation rather than of the result: the exact form of the condition, which is what the proof actually uses, is cleared under both readings, and every variant in the margin table there is cleared by a factor of at least 1.9 under the token-count reading, including the worst case $\rho_h^{\mathrm{flat}} = 0$ evaluated at the asymptotic design effect.

H.2 Score decomposition and correlation parameters

Every rollout's score decomposes along the generation boundary between the plan segment and the code segment,

$$h^{\mathrm{flat}}_{i,j} = h^{\mathrm{plan}}_{i} + h^{\mathrm{cond}}_{i,j}, \qquad h^{\mathrm{plan}}_{i} := \nabla_\theta \log \pi_\theta(p_i \mid x), \qquad h^{\mathrm{cond}}_{i,j} := \nabla_\theta \log \pi_\theta(y_{i,j} \mid x, p_i), \tag{55}$$

with h^{plan}_i identical across all rollouts sharing plan i. GRPO differentiates the unconditioned likelihood and therefore carries h^{flat}; PROSE's code-level term differentiates the plan-conditioned likelihood and carries h^{cond}. Define the two within-plan score-correlation parameters

$$\rho_h^{\mathrm{flat}} := \frac{1}{kM(M-1)\|h^{\mathrm{flat}}\|^2} \sum_{i=1}^{k} \sum_{j \neq j'} \big\langle h^{\mathrm{flat}}_{i,j},\, h^{\mathrm{flat}}_{i,j'} \big\rangle \qquad \text{(unconditioned, used by GRPO)}, \tag{56}$$

$$\rho_h^{\mathrm{cond}} := \frac{1}{kM(M-1)\|h^{\mathrm{cond}}\|^2} \sum_{i=1}^{k} \sum_{j \neq j'} \big\langle h^{\mathrm{cond}}_{i,j},\, h^{\mathrm{cond}}_{i,j'} \big\rangle \qquad \text{(plan-conditioned, used by PROSE)}, \tag{57}$$

both understood in the expected sense of Equation (54).

The two scores have different norms: the common shorthand $\|h\|^2$ conflates them and silently inflates the PROSE side of the bound.

Remark H.2 (Orthogonality of the plan and code score components). The cross-term between the plan and code score components is exactly zero, not merely small. Fix (x, p_i). The score-function identity $\mathbb{E}_{y\sim\pi_\theta(\cdot|x,p_i)}[\nabla_\theta \log \pi_\theta(y \mid x, p_i)] = 0$ holds for every θ (it is $\nabla_\theta \int \pi_\theta = \nabla_\theta 1 = 0$, not an equilibrium condition), and h_i^{plan} is measurable with respect to (x, p_i), hence constant under that expectation. Therefore

$$\mathbb{E}\big\langle h_i^{\text{plan}},\, h_{i,j}^{\text{cond}} \big\rangle = \mathbb{E}_{x,p_i}\Big\langle h_i^{\text{plan}},\, \mathbb{E}_{y_{i,j}}\big[h_{i,j}^{\text{cond}} \;\big|\; x, p_i\big]\Big\rangle = 0. \tag{58}$$

Expanding Equation (55) for $j \neq j'$ and taking expectations, both cross-terms vanish by Equation (58) and the identity collapses to two clean statements:

$$\big\langle h_{i,j}^{\text{flat}}, h_{i,j'}^{\text{flat}} \big\rangle = \|h^{\text{plan}}\|^2 + \big\langle h_{i,j}^{\text{cond}}, h_{i,j'}^{\text{cond}} \big\rangle, \qquad \|h^{\text{flat}}\|^2 = \|h^{\text{plan}}\|^2 + \|h^{\text{cond}}\|^2. \tag{59}$$

Two consequences are used below: $\|h^{\text{flat}}\|^2 \geq \|h^{\text{cond}}\|^2$, and the cross-level orthogonality Equation (58) that enters Equation (65). A third is used only as a consistency check on the measurements: since ρ_h^{flat} is the weighted average of 1 and ρ_h^{cond} with weights $\|h^{\text{plan}}\|^2$ and $\|h^{\text{cond}}\|^2$, we always have $\rho_h^{\text{flat}} \geq \rho_h^{\text{cond}}$, the ordering the proxy measurements confirm (0.56 versus 0.21, Table 12) — though that same weighting pins the score-norm ratio κ to a value the token-count proxy does not reproduce (Appendix H.7). Because Equation (54) defines every one of these quantities as an expectation, the identities in Equation (59) are exact at the level at which they are applied, and no finite-sample concentration argument is required; Equation (59) is a Pythagorean identity, and $\gamma^{\text{flat}} = \gamma^{\text{cond}} + \gamma^{\text{plan}}$ below is just a restatement of it. □

H.3 The one-step improvement lemma

We use the textbook one-step descent lemma for L-smooth J and unbiased $\hat{g}$ (Karimi et al., 2016; Liu et al., 2022): the update $\theta' = \theta + \alpha\hat{g}$ satisfies $\mathbb{E}[J(\theta')] \geq J(\theta) + \alpha\|g\|^2 - \frac{L\alpha^2}{2}(\|g\|^2 + \Sigma)$. Optimising over α at $\alpha^* = \|g\|^2/(L(\|g\|^2 + \Sigma))$ gives

$$\Delta J^* \;=\; \frac{\|g\|^2}{2L} \cdot \frac{1}{1 + \Sigma/\|g\|^2}, \qquad \frac{\Delta J_1^*}{\Delta J_2^*} \;=\; \frac{1 + \Sigma_2/\|g\|^2}{1 + \Sigma_1/\|g\|^2}. \tag{60}$$

Comparing per-step improvement therefore reduces entirely to comparing the two noise-to-signal ratios $\Sigma/\|g\|^2$. Note that Equation (60) is monotone decreasing in Σ, which is the property the iteration-complexity corollary inherits (Appendix J).

H.4 GRPO gradient variance

GRPO uses the flat advantage $A_{i,j}^{\text{flat}} = r_{i,j} - \bar{r}$, whose covariance structure is Lemma G.2. Under (A1) the cross-plan terms drop out of Equation (53) because the *scores* are orthogonal, not because the covariances are small, so no approximation is incurred there. Writing $s_d := \text{Var}(\widehat{A}_{i,j}^{\text{flat}})/\sigma^2$ and $s_w := \text{Cov}(\widehat{A}_{i,j}^{\text{flat}}, \widehat{A}_{i,j'}^{\text{flat}})/\sigma^2$ for the diagonal and same-plan entries of Equation (47), and using $kM(M-1) = N(M-1)$,

$$\begin{aligned}\Sigma^{\text{GRPO}} &= \frac{1}{N^2}\Bigg[\underbrace{\sum_{(i,j)} \|h_{i,j}^{\text{flat}}\|^2\, \sigma^2 s_d}_{\text{diagonal}} + \underbrace{\sum_i \sum_{j\neq j'} \big\langle h_{i,j}^{\text{flat}}, h_{i,j'}^{\text{flat}} \big\rangle\, \sigma^2 s_w}_{\text{within-plan off-diagonal}}\Bigg] \\ &= \frac{\|h^{\text{flat}}\|^2\sigma^2}{N}\Big[\, s_d + (M-1)\,\rho_h^{\text{flat}}\, s_w \,\Big] \;=:\; \frac{\|h^{\text{flat}}\|^2\sigma^2}{N}\, D_{\text{eff}}^{\text{batch}}.\end{aligned} \tag{61}$$

Letting the number of plans grow with M fixed, $s_d \to 1$ and $s_w \to \rho_r$, and Equation (61) takes the form quoted in the main text,

$$\Sigma^{\text{GRPO}} \;=\; \frac{\|h^{\text{flat}}\|^2\sigma^2}{N}\big[1 + (M-1)\rho_h^{\text{flat}}\rho_r\big] \;=:\; \frac{\|h^{\text{flat}}\|^2\sigma^2}{N}\, D_{\text{eff}}^{\text{GRPO}}. \tag{62}$$

This is the correct form of the design effect at the gradient level: the *product* $\rho_h^{\text{flat}}\rho_r$ replaces ρ_r alone, because gradient inflation requires both reward correlation and score correlation within the plan. If

either vanishes the within-plan off-diagonal contribution collapses and $D_{\text{eff}} \to 1$. Setting $\rho_h^{\text{flat}} = 1$ recovers the reward-level design effect $D_r = 1 + (M-1)\rho_r$ as an upper bound; setting $\rho_h^{\text{flat}} = 0$ gives the rollout-orthogonal lower bound 1. Remark H.4 quantifies the gap between Equation (61) and Equation (62) at our operating point and confirms that using the latter is the GRPO-favourable choice we must nonetheless defend.

H.5 PROSE GRADIENT VARIANCE

Code-level term. PROSE's code-level advantage $A_{i,j}^{\text{code}} = r_{i,j} - \widehat{V}_i$ cancels the plan effect exactly (Lemma G.1), and its covariance structure in Equation (48) is exact; no $O(1/N)$ terms are dropped on this side. Substituting the plan-conditioned scores,

$$\begin{aligned}\Sigma_{\text{code}}^{\text{PROSE}} &= \frac{1}{N^2}\Big[N\,\mathbb{E}\|h^{\text{cond}}\|^2 \cdot \tfrac{M-1}{M}\sigma_W^2 - N(M-1)\,\rho_h^{\text{cond}}\,\mathbb{E}\|h^{\text{cond}}\|^2 \cdot \tfrac{\sigma_W^2}{M}\Big] \\ &= \frac{\mathbb{E}\|h^{\text{cond}}\|^2\,\sigma_W^2\,(M-1)}{NM}\big[1-\rho_h^{\text{cond}}\big] \;\le\; \frac{\mathbb{E}\|h^{\text{cond}}\|^2\,(1-\rho_r)\,\sigma^2}{N}. \end{aligned} \tag{63}$$

The final inequality discards two factors that both favour PROSE, namely $(M-1)/M < 1$ and $1 - \rho_h^{\text{cond}} < 1$ under (A2) applied to the conditioned scores, yielding a clean upper bound. The norm $\|h^{\text{cond}}\|^2$ here is strictly smaller than GRPO's $\|h^{\text{flat}}\|^2$ by Equation (59), because plan conditioning removes the common plan-prefix component from the carrier of the gradient signal.

Plan-level term. The plan-level term of $\mathcal{L}_{core}$ contributes a second gradient with score h_i^{plan} and advantage $\widehat{A}_{\text{plan},i} = \widehat{V}_i - \bar{V}$, averaged over k rather than N terms. Under balanced allocation ($\omega_i \equiv 1$) and (A1) applied to $\{h_i^{\text{plan}}\}$, Equation (45) gives $\text{Var}(\widehat{A}_{\text{plan},i}) = \frac{k-1}{k}(\sigma_B^2 + \sigma_W^2/M)$, in which the second component is the estimation noise incurred by using M samples per plan mean. Bounding $\frac{k-1}{k} \le 1$ and using $1/k = M/N$,

$$\begin{aligned}\Sigma_{\text{plan}}^{\text{PROSE}} \;&\le\; \frac{\mathbb{E}\|h^{\text{plan}}\|^2\big(\sigma_B^2 + \sigma_W^2/M\big)}{k} \;=\; \frac{M\,\mathbb{E}\|h^{\text{plan}}\|^2\sigma^2}{N}\left(\rho_r + \frac{1-\rho_r}{M}\right) \\ &=\; \frac{\mathbb{E}\|h^{\text{plan}}\|^2\sigma^2}{N}\big[1 + (M-1)\rho_r\big]. \end{aligned} \tag{64}$$

The last step is an identity, not a further bound: $M\rho_r + (1-\rho_r) = 1 + (M-1)\rho_r = D_r$. The plan-level term therefore carries the full reward-level design effect: PROSE pays D_r on the k-dimensional plan channel in exchange for removing it from the N-dimensional code channel, and the trade is favourable precisely when the plan channel carries a small share of the score norm. That is the content of (A3).

Total. Cross-level orthogonality Equation (58) makes the two channels uncorrelated, so

$$\Sigma^{\text{PROSE}} \;=\; \Sigma_{\text{code}}^{\text{PROSE}} + \Sigma_{\text{plan}}^{\text{PROSE}} \;\le\; \frac{\sigma^2}{N}\Big[\mathbb{E}\|h^{\text{cond}}\|^2(1-\rho_r) + \mathbb{E}\|h^{\text{plan}}\|^2\big(1+(M-1)\rho_r\big)\Big]. \tag{65}$$

H.6 COMBINING: WHERE (A3) ENTERS

Define the method-specific noise-to-signal ratios

$$\gamma^{\text{flat}} := \frac{\|h^{\text{flat}}\|^2\sigma^2}{N\|g\|^2}, \quad \gamma^{\text{cond}} := \frac{\|h^{\text{cond}}\|^2\sigma^2}{N\|g\|^2}, \quad \gamma^{\text{plan}} := \frac{\|h^{\text{plan}}\|^2\sigma^2}{N\|g\|^2}, \tag{66}$$

so that $\gamma^{\text{flat}} = \gamma^{\text{cond}} + \gamma^{\text{plan}}$ exactly, by Equation (59). Then $\Sigma^{\text{GRPO}}/\|g\|^2 = \gamma^{\text{flat}} D_{\text{eff}}^{\text{GRPO}}$ and, by Equation (65), $\Sigma^{\text{PROSE}}/\|g\|^2 \le \gamma^{\text{cond}}(1-\rho_r) + \gamma^{\text{plan}}(1 + (M-1)\rho_r)$. Substituting into Equation (60),

$$\frac{\Delta J^{\text{PROSE}}}{\Delta J^{\text{GRPO}}} \;\ge\; \frac{1 + \gamma^{\text{flat}} D_{\text{eff}}^{\text{GRPO}}}{1 + \gamma^{\text{cond}}(1-\rho_r) + \gamma^{\text{plan}}(1 + (M-1)\rho_r)} \;\ge\; 1. \tag{67}$$

The first inequality is Equation (60) together with the two variance bounds. The second is the step that requires (A3), and it is worth doing in one line rather than by a chain of pairwise comparisons,

because the termwise route is where the previously implicit condition becomes visible. Expanding the numerator minus the denominator of the middle expression in Equation (67) and collecting terms in γ^{cond} and γ^{plan},

$$\begin{aligned}\gamma^{\mathrm{flat}} D_{\mathrm{eff}}^{\mathrm{GRPO}} &- \gamma^{\mathrm{cond}}(1-\rho_r) - \gamma^{\mathrm{plan}}\big(1+(M{-}1)\rho_r\big) \\ &= \gamma^{\mathrm{cond}}\Big[1+(M{-}1)\rho_h^{\mathrm{flat}}\rho_r - 1 + \rho_r\Big] + \gamma^{\mathrm{plan}}\Big[1+(M{-}1)\rho_h^{\mathrm{flat}}\rho_r - 1 - (M{-}1)\rho_r\Big] \\ &= \rho_r\Big\{\gamma^{\mathrm{cond}}\big[1+(M{-}1)\rho_h^{\mathrm{flat}}\big] - \gamma^{\mathrm{plan}}(M{-}1)\big(1-\rho_h^{\mathrm{flat}}\big)\Big\}. \end{aligned} \tag{68}$$

Note what Equation (68) does not require: nowhere does the argument compare $1+(M{-}1)\rho_h^{\mathrm{flat}}\rho_r$ against $1+(M{-}1)\rho_r$ termwise, a comparison that would need $\rho_h^{\mathrm{flat}} \geq 1$ and is therefore unavailable for a correlation. The γ^{cond} and γ^{plan} channels must be compared jointly, and the sum is what is non-negative.

Since $\rho_r > 0$ and $\gamma^{\bullet} \propto \|h^{\bullet}\|^2$, the braced quantity in Equation (68) is non-negative exactly when

$$\big[1+(M{-}1)\rho_h^{\mathrm{flat}}\big]\,\|h^{\mathrm{cond}}\|^2 \;\geq\; (M{-}1)\big(1-\rho_h^{\mathrm{flat}}\big)\,\|h^{\mathrm{plan}}\|^2, \tag{69}$$

which is the exact form of (A3). The sufficient form stated in Assumption H.1 implies Equation (69) because $1+(M{-}1)\rho_h^{\mathrm{flat}} \geq 1$ under (A2); the exact form is weaker by exactly that factor. This proves Equation (67), with strictness whenever $\rho_r > 0$ and Equation (69) holds strictly. □

H.7 Empirical status of (A3)

It is convenient to state the requirement as a single threshold on the score-norm ratio $\kappa := \|h^{\mathrm{cond}}\|^2/\|h^{\mathrm{plan}}\|^2$. For any GRPO-side design effect $D > 1-\rho_r$, the condition $\gamma^{\mathrm{flat}} D \geq \gamma^{\mathrm{cond}}(1-\rho_r) + \gamma^{\mathrm{plan}}(1+(M{-}1)\rho_r)$ is equivalent to

$$\kappa \;\geq\; \tau(D) := \frac{1+(M{-}1)\rho_r - D}{D-1+\rho_r}. \tag{70}$$

Evaluating Equation (70) at $D = D_{\mathrm{eff}}^{\mathrm{GRPO}} = 1+(M{-}1)\rho_h^{\mathrm{flat}}\rho_r$ recovers Equation (69) and, notably, ρ_r cancels:

$$\tau\big(D_{\mathrm{eff}}^{\mathrm{GRPO}}\big) = \frac{(M{-}1)\rho_r(1-\rho_h^{\mathrm{flat}})}{\rho_r\big[1+(M{-}1)\rho_h^{\mathrm{flat}}\big]} = \frac{(M{-}1)(1-\rho_h^{\mathrm{flat}})}{1+(M{-}1)\rho_h^{\mathrm{flat}}}, \tag{71}$$

so the exact threshold depends only on M and the score correlation, not on the reward ICC.

The proxy value of κ, and a disagreement we do not resolve. We do not measure κ directly: computing $\nabla_\theta \log \pi_\theta$ per rollout is prohibitive at 7B scale (Appendix L). We use the paper's norm heuristic under a proportional-to-token-count reading: a plan occupies $\sim$35 tokens against $\sim$200+ code tokens, giving $\kappa \approx 200/35 \approx 5.7$. This is an order-of-magnitude proxy, and it does not agree with the second route to κ that our own measurements open. Equation (59) makes ρ_h^{flat} the weighted average of 1 and ρ_h^{cond} with weights $\|h^{\mathrm{plan}}\|^2$ and $\|h^{\mathrm{cond}}\|^2$, i.e. $(1+\kappa)\rho_h^{\mathrm{flat}} = 1+\kappa\,\rho_h^{\mathrm{cond}}$ exactly, so the cosine proxies $\widehat{\rho}_h^{\mathrm{flat}} = 0.56$ and $\widehat{\rho}_h^{\mathrm{cond}} = 0.21$ of Table 12 imply

$$\kappa \;=\; \frac{1-\rho_h^{\mathrm{flat}}}{\rho_h^{\mathrm{flat}} - \rho_h^{\mathrm{cond}}} \;=\; \frac{0.44}{0.35} \;\approx\; 1.26, \tag{72}$$

a plan prefix carrying $\approx$44% of the squared score norm rather than the $\approx$15% the token count suggests. Equivalently, $\kappa \approx 5.7$ together with $\widehat{\rho}_h^{\mathrm{cond}} = 0.21$ would force $\rho_h^{\mathrm{flat}} \approx 0.33$, not 0.56. The two proxies measure different objects — a hidden-state cosine and a token count — and are not calibrated against each other; nor is Equation (56), a ratio of averaged inner products, the same functional as the averaged per-pair cosine that Table 12 reports, so Equation (59) need not close on the estimates even though it is exact for the quantities it is stated for. We do not reconcile them, and we report Table 10 under both readings rather than choosing the favourable one.

Three readings of Table 10 are worth stating plainly. (i) The condition the proof actually needs, Equation (69), is the only variant cleared under both readings of κ: by $11.6\times$ and $2.6\times$ at the asymptotic design effect, and by $5.4\times$ and $1.2\times$ at the finite-batch one. (ii) The stated sufficient form (A3) is cleared by $4.3\times$ under the token-count reading, which is the margin quoted in the

Table 10: Thresholds on the score-norm ratio $\kappa = \|h^{\text{cond}}\|^2/\|h^{\text{plan}}\|^2$ required for Equation (67), under successively less favourable readings of the comparison. All rows are Equation (70) evaluated at the indicated D, with $M = 4$, $N = 16$, $k = 4$, $\rho_r = 0.72$, $\widehat{\rho}_h^{\text{flat}} = 0.56$, $\widehat{\rho}_{\text{cross}} = 0.08$. Rows 3–4 use the finite-batch design effect $D_{\text{eff}}^{\text{batch}}$ of Equation (61); rows 1 and 5–6 use the asymptotic one. The two margin columns are κ/τ at the token-count proxy $\kappa \approx 5.7$ and at the $\kappa \approx 1.26$ that Equation (72) implies from the measured score correlations; a margin below 1 means the variant is *not* cleared under that reading. The deployed-comparator row is derived in Appendix H.8; the finite-batch rows in Remark H.4.

			Margin κ/τ at	
Variant of the comparison	**GRPO-side D**	**Threshold τ**	$\kappa\approx 5.7$	$\kappa\approx 1.26$
Exact form (69), asymptotic D_{eff}	2.21	0.49	11.6×	2.6×
Sufficient form (A3) as stated	—	1.32	4.3×	0.95×
Exact form, finite batch $D_{\text{eff}}^{\text{batch}}$	1.68	1.06	5.4×	1.2×
Finite batch *and* (A1) relaxed, ρ_{cross}=0.08	1.43	1.50	3.8×	0.84×
Deployed comparator, $D_{\text{eff}} \to 1$	1.00	3.00	1.9×	0.42×
Worst case $\rho_h^{\text{flat}} = 0$ (i.e. (A2) tight)	1.00	3.00	1.9×	0.42×

main text and in Table 1; under the measurement-implied $\kappa \approx 1.26$ it is missed by 5%. (iii) Each unfavourable choice taken on its own — finite batch, relaxed cross-plan orthogonality, or the collapse of the within-plan score correlation to zero — leaves a threshold of at most 3 against a proxy of ≈ 5.7. Rows 5–6 evaluate the last of these at the asymptotic design effect; carrying the finite-batch convention of rows 3–4 into it ($D = s_d = 0.8025$ at $\rho_h^{\text{flat}} = 0$) raises the threshold to 4.51, a margin of $1.3\times$, and stacking the cross-plan correction on top of that ($D = 0.55$) raises it to 9.7, which the proxy does not clear; the deployed comparator at finite batch, $D = (N-1)/N = 0.9375$, likewise gives $\tau = 3.38$ and $1.7\times$. We claim the ordering for the variants tabulated, not for every combination of them.

What the two readings of κ agree on is the exact form Equation (69), which is what the proof of Equation (67) actually uses. Part (c) should therefore be read as resting on that form against the counterfactually clustered comparator the theorem is stated for; its sufficient-form and deployed-comparator variants are supported only under the token-count reading of κ. The gradient-level claim was in this sense never at risk from the omission of (A3); what was at risk was the honesty of the assumption list, which is why (A3) now appears in the statement rather than inside the algebra.

H.8 Which GRPO is being compared against

D_{eff} is a *same-batch* quantity: it presumes the N rollouts are organised as k plans of M, which is PROSE's organisation, not GRPO's. Part (c) and the iteration-complexity corollary therefore compare against the **counterfactually clustered** GRPO gradient, exactly as Equation (49) does at the reward level. The **deployed** analogue — GRPO as actually run, i.i.d. with no within-batch plan structure — sets $D_{\text{eff}} \to 1$, and repeating the step of Equation (68) with $D = 1$ gives the requirement

$$\gamma^{\text{cond}} \;\geq\; (M-1)\,\gamma^{\text{plan}}, \qquad \text{i.e.} \qquad \kappa \;\geq\; M - 1 = 3 \tag{73}$$

at $M = 4$, which is the last row of Table 10: cleared under the token-count reading of κ, not under the measurement-implied one. Equation (73) coincides with Equation (69) evaluated at $\rho_h^{\text{flat}} = 0$, which is the right reading: a deployed i.i.d. batch has no within-plan structure for the score correlation to act on. Both comparators are labelled wherever either appears (in Theorem 4.1(a) and (c), in Corollary J.1, and in Table 9), so that a clustered prediction is never paired against a deployed measurement or vice versa.

H.9 SPARSE-REWARD LIMIT AND REGIME CHECK

As the noise-to-signal ratios diverge (the sparse-reward regime in which execution feedback is rare and $\|g\|^2$ is small relative to σ^2), the constants in Equation (67) become negligible and

$$\frac{1+\gamma^{\text{flat}} D_{\text{eff}}^{\text{GRPO}}}{1+\gamma^{\text{cond}}(1-\rho_r)+\gamma^{\text{plan}}(1+(M-1)\rho_r)} \longrightarrow \frac{\|h^{\text{flat}}\|^2\big(1+(M-1)\rho_h^{\text{flat}}\rho_r\big)}{\|h^{\text{cond}}\|^2(1-\rho_r)+\|h^{\text{plan}}\|^2\big(1+(M-1)\rho_r\big)}, \tag{74}$$

which combines the reward-level cluster design effect with the score-norm ratio that distinguishes the two estimators' carriers of gradient signal. This is the regime execution-based code RL operates in, and it is where the ratio is largest.

Qualitative regime check. The measured $\rho_r \approx 0.72$ on the APPS+ holdout (Appendix K) and the positive proxy $\widehat{\rho}_h^{\text{flat}} \approx 0.56$ (Table 12) place our 7B setting in the regime where neither the numerator nor the denominator of Equation (74) degenerates, so the per-step comparison is a strict improvement rather than a knife-edge equality. We deliberately stop short of converting Equation (74) into a single point-estimate multiplier on pass@1: the score-norm inputs are proxies, and the one quantitative prediction we do commit to is the advantage-variance ratio of Appendix G.5 (0.262 predicted against 0.26 measured), which requires no gradient-level quantity at all.

H.10 RELAXING THE ASSUMPTIONS

Remark H.3 (Sign and size of the cross-plan term when (A1) is relaxed). Rollouts for the same problem share its description, so distinct plans do not have exactly orthogonal scores. Capture the residual similarity by the normalised quantity

$$\rho_{\text{cross}} \;:=\; \frac{1}{k(k-1)M^2\,\|h^{\text{flat}}\|^2}\sum_{i\neq i'}\sum_{j,j'}\mathbb{E}\big\langle h_{i,j}^{\text{flat}},\, h_{i',j'}^{\text{flat}}\big\rangle \;\geq\; 0,$$

non-negative under positive prefix overlap. The normalisation is by the $k(k-1)M^2$ cross-plan pairs actually summed, so that ρ_{cross} is a mean cosine on the same footing as ρ_h^{flat} in Equation (56). The natural reading, that a positive ρ_{cross} adds a further inflation to GRPO's gradient variance and therefore erodes Equation (67), has the wrong sign. GRPO's flat advantage is centred on the grand mean, which forces $\sum_{i,j}\widehat{A}_{i,j}^{\text{flat}} = 0$ and hence makes the off-diagonal covariances negative on average; Equation (47) gives the cross-plan entry exactly as $-\sigma_B^2/k - \sigma_W^2/N$. Multiplying a non-negative score similarity by a negative advantage covariance gives a negative contribution:

$$\Sigma^{\text{GRPO}} \;=\; \frac{\|h^{\text{flat}}\|^2\sigma^2}{N}\Big[\,D_{\text{eff}}^{\text{batch}} \;-\; \tfrac{k-1}{k}\,\rho_{\text{cross}}\,D_r \;+\; o(\rho_{\text{cross}})\,\Big], \qquad D_r = 1+(M-1)\rho_r, \tag{75}$$

to first order in ρ_{cross}, since $N(\sigma_B^2/k+\sigma_W^2/N)/\sigma^2 = M\rho_r+(1-\rho_r) = D_r$. The coefficient is the full reward-level design effect D_r, not $(M-1)\rho_r$: at the operating point $D_r = 3.16$ and $\widehat{\rho}_{\text{cross}} \approx 0.08$ (Table 12), so the correction is $\frac{3}{4}(0.08)(3.16) = 0.19$ in design-effect units. Throughout Table 10 we nonetheless apply the looser $\rho_{\text{cross}}D_r = 0.25$, dropping the $\frac{k-1}{k}$ factor: that over-applies the correction by $4/3$ and so errs against us, by the same convention under which we discard every PROSE-favourable factor elsewhere in this appendix. The looser value is about 11% of $D_{\text{eff}} = 2.21$.

The correction runs against us, and we keep it. PROSE's code-level advantage has exactly zero cross-plan covariance (Equation (48)), so no cross-plan score similarity can reach it; its plan-level term does acquire an analogous negative contribution, which would reduce Σ^{PROSE} in turn, and we discard that benefit. The net accounting in Table 10 therefore applies the relaxation only where it hurts. The honest conclusion is the opposite of "ρ_{cross} is negligible": at the measured value it removes 0.25 from the GRPO-side design effect (11% of the asymptotic $D_{\text{eff}} = 2.21$ and 15% of the finite-batch $D_{\text{eff}}^{\text{batch}} = 1.68$), raising the (A3) threshold from 1.06 to 1.50, and the ordering survives with a $3.8\times$ margin at the token-count κ — though not at the measurement-implied $\kappa \approx 1.26$ (Table 10); the tighter coefficient would give 0.19, $D = 1.49$ and a threshold of 1.38 instead. The dominant route by which the bound could actually fail is not ρ_{cross} but erosion of the cross-level orthogonality Equation (58), which Equation (54) makes exact under within-plan sampling but which a stale-policy or off-policy correction would perturb. □

Remark H.4 (Finite-batch accounting, and why the asymptotic D_{eff} is not free). Equation (62) drops terms of order $1/k$ and $1/N$ relative to Equation (61). Those terms are not decorative: they are precisely what enforces $\sum_{i,j} \widehat{A}^{\text{flat}}_{i,j} = 0$ (Lemma G.2), and discarding them overstates the GRPO-side variance. At the operating point $k = M = 4$, $N = 16$, $\rho_r = 0.72$, $\rho_h^{\text{flat}} = 0.56$,

$$s_d = 0.8025, \qquad s_w = 0.5225, \qquad D_{\text{eff}}^{\text{batch}} = 0.8025 + 3(0.56)(0.5225) = 1.68,$$

against an asymptotic $D_{\text{eff}}^{\text{GRPO}} = 1 + 3(0.56)(0.72) = 2.21$. The asymptotic form is thus about a third larger than the finite-batch value that a 16-rollout group actually realises. Adding the cross-plan correction of Equation (75) brings the finite-batch design effect to $1.68 - 0.25 = 1.43$. We report $D_{\text{eff}} = 2.21$ in the main text because it is the limit the theorem is stated in and because the budget-curve translation it feeds is explicitly a secondary sanity check rather than a theorem-level prediction; that choice is not free, and we state its cost. The finite-batch value translates to $\log_2(1.68) \approx 0.75$ budget doublings and hence $\approx +1.5$ pass@1, *below* the $+1.9$ observed rather than just above it as the $+2.2$ from 2.21 is. What that check tests — that the gradient-level route sits below the reward-level route, which brackets the observation from above at $+3.2$ — holds under either value; the closeness of $+2.2$ to $+1.9$ does not, and we do not claim it as evidence. The comparison that matters, whether Equation (67) holds, is verified against the smaller finite-batch values in rows 3 and 4 of Table 10, where it still clears by $3.8\times$ at the token-count κ and, for row 3, by $1.2\times$ at the measurement-implied one. The corresponding finite-batch effect on the PROSE side is not claimed as an offsetting benefit: both Equation (63) and Equation (64) are already upper bounds that discard their own $(M-1)/M$, $1 - \rho_h^{\text{cond}}$ and $(k-1)/k$ factors. □

Remark H.5 (Scope). Three limits on what this appendix establishes. (i) The result is a *relative* per-step statement between two optimally-stepped estimators under an L-smoothness assumption that we do not verify for a 7B transformer; it is not an absolute improvement guarantee. (ii) Conventions (C1)–(C2) are modelling choices; in particular the score–advantage decoupling of (C2) is an approximation applied to both sides. (iii) ρ_h^{flat}, ρ_h^{cond} and κ are all reached through proxies (hidden-state cosine, token counts), never measured directly, and the two routes to κ disagree by a factor of ≈ 4.5 (Appendix H.7), so part (c) should be read as design rationale with a checkable assumption set, while the falsifiable quantitative prediction of the theory remains the advantage-variance ratio at the reward level (Appendix G.5). □

I Optimality of the Hierarchical Advantage: UMVU Within the Symmetric Subtractive Class

This appendix proves the optimality half of Theorem 4.1(b). The statement is a *conditional* optimality claim: the hierarchical advantage is uniformly minimum-variance unbiased (UMVU) *within the symmetric subtractive class* defined below, and not among all conceivable advantage estimators. The qualifier is load-bearing and we carry it at every mention. Estimators that consume finer-grained signal than a per-rollout scalar execution reward—token-level process rewards, per-test partial credit, intermediate execution traces (Ye et al., 2025)—leave this class, and the bound of Equation (77) does not apply to them. Our setting supplies exactly one scalar per rollout ($r(y) \in \{0, 1\}$, Section 3), which is what makes the class the natural one here; it is a scope boundary, not a claim of universal optimality.

Proposition I.1 (UMVU within the symmetric subtractive class)**.** *Adopt the one-way random-effects model of Appendix G and define the class of plan-membership-based advantage estimators in symmetric subtractive form*

$$\mathcal{F} := \left\{ \widehat{A}_{i,j} = r_{i,j} - b\big(\{r_{i,m}\}_{m=1}^{M_i}\big) \;\middle|\; b \text{ symmetric measurable},\ \mathbb{E}[b \mid \alpha_i] = v_i \right\}, \tag{76}$$

i.e., a baseline that is permutation-invariant in the within-plan reward multiset and plan-conditionally unbiased for $v_i := \mu + \alpha_i$. Symmetry encodes the plan-membership access pattern (the estimator sees which rollouts share a plan, and nothing finer); plan-conditional unbiasedness is the condition under which the REINFORCE update is unbiased for the plan-conditional gradient target. Assume in addition the regularity condition

- ***(C)*** *the plan-conditional reward law $r_{i,m} \mid \alpha_i \sim P_{v_i}$ belongs to a one-parameter exponential family whose mean parameter v_i ranges over a set containing an open interval.*

Then PROSE*'s code-level advantage* $\widehat{A}_{\mathrm{code},i,j} = r_{i,j} - \widehat{V}_i$*, with* $\widehat{V}_i = M_i^{-1} \sum_m r_{i,m}$*, is the UMVU element of* $\mathcal{F}$*, with second moment*

$$\inf_{\widehat{A} \in \mathcal{F}} \mathbb{E}\Big[\widehat{A}_{i,j}^2\Big] \;=\; \frac{M_i - 1}{M_i}\,\sigma_W^2, \tag{77}$$

attained uniquely (a.s.) by $\widehat{A}_{\mathrm{code},i,j}$*. Consequently the variance-reduction fraction of Equation* (82) *below is the maximum achievable by any estimator in* $\mathcal{F}$ *relative to the flat grand-mean baseline.*

Both reward families that the paper's moment model covers satisfy (C): Bernoulli rewards (the execution-based setting we study, $r \in \{0, 1\}$) with $v_i \in (0, 1)$, and Gaussian rewards with known within-plan variance σ_W^2 and $v_i \in \mathbb{R}$. Condition (C) is what makes Lehmann–Scheffé available. It is an added assumption, not a consequence of the model of Appendix G.1: that model fixes second moments and no distributional form at all, whereas (C) names the plan-conditional family and, separately, requires the plan values v_i to range over a set containing an open interval. The second requirement is what completeness actually needs, and non-degeneracy of the mixing law of α_i does not supply it: a mixing law supported on finitely many values of v_i is non-degenerate, but the family indexed by that finite set is incomplete and the step at Equation (80) fails outright. We therefore assume (C), noting that it holds for any mixing law whose support contains an interval.

Setup and notation. Under the one-way random-effects model, $r_{i,j}$ has plan-conditional mean $v_i = \mu + \alpha_i$ and $\{r_{i,m}\}_{m=1}^{M_i}$ are conditionally i.i.d. given α_i. Write $s^2(\alpha_i) := \mathrm{Var}(r \mid \alpha_i)$ for the plan-conditional variance, so that $\sigma_W^2 = \mathbb{E}[s^2(\alpha_i)]$, $\sigma_B^2 = \mathrm{Var}(v_i)$, $\sigma^2 = \sigma_B^2 + \sigma_W^2$ and $\rho_r = \sigma_B^2/\sigma^2$. In the Bernoulli case $s^2(\alpha_i) = v_i(1 - v_i)$. All steps below are carried out conditionally on α_i and then averaged over the plan population, so the conclusion is uniform in α_i and does not depend on the mixing distribution.

Step 1: completeness of the within-plan sample mean. Fix α_i. Under (C) the plan-conditional model $r_{i,1}, \ldots, r_{i,M_i} \mid \alpha_i \overset{\text{iid}}{\sim} P_{v_i}$ is a one-parameter exponential family in v_i, and $\widehat{V}_i$ is its natural sufficient statistic:

- **Bernoulli.** The within-plan total $M_i\widehat{V}_i$ is $\mathrm{Binomial}(M_i, v_i)$, an exponential family whose natural parameter $\log\big(v_i/(1 - v_i)\big)$ ranges over an open subset of $\mathbb{R}$; it is complete for $v_i \in (0, 1)$.
- **Gaussian with known** σ_W^2**.** Here $\widehat{V}_i \sim \mathcal{N}(v_i, \sigma_W^2/M_i)$, so the within-plan total $M_i\widehat{V}_i$ is an exponential family with natural parameter v_i/σ_W^2, ranging over all of $\mathbb{R}$; completeness again follows.

In both cases $\widehat{V}_i$ is a complete sufficient statistic for v_i (Lehmann & Romano, 2005, Thm. 4.3.1).

We stress what is not used. It is tempting to argue that a symmetric measurable b factors through the order statistic of $\{r_{i,m}\}$, and that the order statistic generates the same σ-algebra as $\widehat{V}_i$. The second half of that argument is false in general: for M_i distinct reals the order statistic strictly refines $\sigma(\widehat{V}_i)$, so the equivalence holds for Bernoulli (where the order statistic and the sum carry identical information) but fails for Gaussian. The proof below therefore never invokes order-statistic equivalence; it applies Lehmann–Scheffé directly to the complete sufficient statistic $\widehat{V}_i$, which is valid for both families. Symmetry of b is still used, but for a different and precise purpose, isolated in Step 2.

Step 2: an exact second-moment decomposition, with a class-invariant cross term. Let $\widehat{A}_{i,j} = r_{i,j} - b \in \mathcal{F}$ and write $\varepsilon_{i,j} := r_{i,j} - v_i$ and $\delta := v_i - b$. Since $\mathbb{E}[b \mid \alpha_i] = v_i$ we have $\mathbb{E}[\widehat{A}_{i,j} \mid \alpha_i] = 0$, so second moment and conditional variance coincide. Expanding $\widehat{A}_{i,j} = \varepsilon_{i,j} + \delta$,

$$\mathbb{E}\big[\widehat{A}_{i,j}^2 \;\big|\; \alpha_i\big] = \underbrace{s^2(\alpha_i)}_{\text{fixed}} - 2\,\underbrace{\mathrm{Cov}\big(r_{i,j},\, b \;\big|\; \alpha_i\big)}_{\text{cross term}} + \underbrace{\mathrm{Var}\big(b \;\big|\; \alpha_i\big)}_{\text{estimation error of } b}. \tag{78}$$

The first term does not depend on b. The third term is exactly the conditional mean squared error of b as an estimator of v_i. The cross term is where symmetry enters. Because b is permutation-invariant in

$\{r_{i,m}\}_{m=1}^{M_i}$ and the rewards are conditionally exchangeable given α_i, the quantity $\mathrm{Cov}(r_{i,m}, b \mid \alpha_i)$ is the same for every m, hence

$$\mathrm{Cov}\big(r_{i,j}, b \bigm| \alpha_i\big) = \frac{1}{M_i}\sum_{m=1}^{M_i} \mathrm{Cov}\big(r_{i,m}, b \bigm| \alpha_i\big) = \mathrm{Cov}\big(\widehat{V}_i,\, b \bigm| \alpha_i\big). \tag{79}$$

This identity is the reason the class is defined with a symmetric baseline: it converts a j-specific covariance into a statistic-level one, and Step 3 shows the right-hand side is the same for every member of $\mathcal{F}$. The minimisation over $\mathcal{F}$ therefore collapses onto the single term $\mathrm{Var}(b \mid \alpha_i)$, which is the classical UMVU problem.

Step 3: Lehmann–Scheffé. Define the Rao–Blackwellisation $b^\star := \mathbb{E}[b \mid \widehat{V}_i, \alpha_i]$. Because $\widehat{V}_i$ is sufficient for v_i, the conditional law of the sample given $\widehat{V}_i$ does not depend on v_i, so $b^\star$ is a genuine statistic (a measurable function of $\widehat{V}_i$ alone, not of the unknown v_i). By the tower property $b^\star$ inherits conditional unbiasedness, $\mathbb{E}[b^\star \mid \alpha_i] = v_i$, and $\widehat{V}_i$ is itself conditionally unbiased for v_i. Completeness of $\widehat{V}_i$ then forces

$$b^\star = \widehat{V}_i \quad \text{a.s.}, \tag{80}$$

since $b^\star - \widehat{V}_i$ is a function of $\widehat{V}_i$ with zero conditional mean for every v_i in an open interval. Two consequences follow.

(i) The cross term is class-invariant. By the tower property and Equation (80),

$$\mathrm{Cov}\big(\widehat{V}_i, b \bigm| \alpha_i\big) = \mathrm{Cov}\big(\widehat{V}_i,\, \mathbb{E}[b \mid \widehat{V}_i, \alpha_i] \bigm| \alpha_i\big) = \mathrm{Cov}\big(\widehat{V}_i, \widehat{V}_i \bigm| \alpha_i\big) = \mathrm{Var}\big(\widehat{V}_i \bigm| \alpha_i\big) = \frac{s^2(\alpha_i)}{M_i},$$

the same value for every b with $\mathbb{E}[b \mid \alpha_i] = v_i$.

(ii) The remaining term is minimised uniquely at $\widehat{V}_i$. By Rao–Blackwell, $\mathrm{Var}(b^\star \mid \alpha_i) \le \mathrm{Var}(b \mid \alpha_i)$ with equality iff $b = b^\star$ a.s.; combined with Equation (80), $\mathrm{Var}(b \mid \alpha_i) \ge \mathrm{Var}(\widehat{V}_i \mid \alpha_i) = s^2(\alpha_i)/M_i$, with equality iff $b = \widehat{V}_i$ a.s. This is the Lehmann–Scheffé conclusion, obtained without any appeal to the order statistic.

Substituting (i) and (ii) into Equation (78),

$$\mathbb{E}\big[\widehat{A}_{i,j}^2 \bigm| \alpha_i\big] \;=\; s^2(\alpha_i) - \frac{2s^2(\alpha_i)}{M_i} + \mathrm{Var}\big(b \bigm| \alpha_i\big) \;\ge\; s^2(\alpha_i)\Big(1 - \frac{2}{M_i} + \frac{1}{M_i}\Big) \;=\; \frac{M_i - 1}{M_i}\, s^2(\alpha_i), \tag{81}$$

with equality iff $b = \widehat{V}_i$ a.s. Because the bound holds pointwise in α_i, it holds after averaging over the plan population for every mixing distribution, giving Equation (77) with $\sigma_W^2 = \mathbb{E}[s^2(\alpha_i)]$, uniformly for the Bernoulli and Gaussian cases. Direct verification at the optimum: writing $r_{i,j} - \widehat{V}_i = \varepsilon_{i,j} - \bar{\varepsilon}_{i\cdot}$, $\mathbb{E}[(r_{i,j} - \widehat{V}_i)^2 \mid \alpha_i] = s^2 - 2s^2/M_i + s^2/M_i = \frac{M_i - 1}{M_i} s^2$, as required. The uniformity in α_i (equivalently, in v_i) is what makes the statement uniformly minimum-variance rather than minimum-variance at a single parameter value. □

Remark I.2 (Symmetry is not a cosmetic restriction). The leave-one-out baseline $b_{-j} = (M_i - 1)^{-1} \sum_{m \ne j} r_{i,m}$ is plan-conditionally unbiased for v_i, but it depends on j and is therefore not an element of $\mathcal{F}$. It also has zero cross term, $\mathrm{Cov}(r_{i,j}, b_{-j} \mid \alpha_i) = 0$, so Equation (78) gives $\mathbb{E}[(r_{i,j} - b_{-j})^2 \mid \alpha_i] = s^2 + s^2/(M_i - 1) = \frac{M_i}{M_i - 1} s^2$, which at $M = 4$ is $\frac{4}{3}s^2$ against $\frac{3}{4}s^2$ for $\widehat{A}_{\mathrm{code}}$, a factor $16/9 \approx 1.8$ worse. The mechanism is visible in Equation (79): the symmetric baseline is deliberately correlated with the very rollout it is subtracted from, and that correlation, fixed at s^2/M_i across the class, is what buys the $(M_i - 1)/M_i$ factor. This is the same trade-off that makes the leave-one-out baseline attractive when strict per-rollout unbiasedness of $\widehat{A}$ given the other rollouts is required, and unattractive when second moment is the objective.

Remark I.3 (What exits the symmetric subtractive class). Equation (77) binds only estimators that access the batch through plan identity and symmetric within-plan aggregates of scalar rollout rewards. Three practically relevant families sit outside it and are not bounded by $(M_i - 1)\sigma_W^2/M_i$: (i) token-level or step-level process rewards, which require a separately trained process reward model (Ye et al., 2025); (ii) partial credit from per-test-case pass fractions, which replaces the Bernoulli $r \in \{0, 1\}$

with a finer-grained signal and thus a different (and lower-variance) conditional model; (iii) learned critics conditioned on the program text rather than on plan membership. PROSE deliberately stays inside the class, because the execution-only regime supplies exactly one scalar per rollout and requires no auxiliary reward model; the price is that our optimality claim is conditional on that regime. We regard combining plan-level clustering with partial-credit rewards as an open direction rather than a contradiction: the clustering argument is orthogonal to the granularity of the reward, but the optimality certificate would have to be re-derived.

I.1 THE VARIANCE-REDUCTION FRACTION AND ITS TWO COMPARATORS

The optimal second moment $\frac{M-1}{M}\sigma_W^2 = \frac{M-1}{M}(1-\rho_r)\sigma^2$ must be compared against a flat grand-mean baseline. The ANOVA identity of Appendix G gives the flat advantage variance on a batch organised as k plans of $M = N/k$ rollouts, $\mathrm{Var}(\widehat{A}^{\mathrm{flat}}_{i,j}) = \frac{k-1}{k}\sigma_B^2 + \frac{N-1}{N}\sigma_W^2$, so the reduction fraction is

$$\Delta^* \;=\; 1 - \frac{\frac{M-1}{M}(1-\rho_r)}{\frac{k-1}{k}\rho_r + \frac{N-1}{N}(1-\rho_r)}. \tag{82}$$

Two instantiations of the comparator answer two different questions and we label them explicitly wherever either appears.

(i) Same-batch comparator. Both estimators are recomputed on one stored clustered batch (k plans of size M). This is the quantity our measurement protocol actually estimates, since the recomputation reuses the logged rollouts. At the training operating point $k = M = 4$, $N = 16$, $\rho_r = 0.72$, Equation (82) evaluates to

$$R_{\mathrm{same}} = \frac{\frac{3}{4}(0.28)}{\frac{3}{4}(0.72) + \frac{15}{16}(0.28)} = \frac{0.2100}{0.8025} = 0.262, \qquad 1 - R_{\mathrm{same}} = 0.738.$$

(ii) GRPO-as-deployed comparator. GRPO as actually run samples i.i.d. without plan conditioning, i.e. $k = N$, $M = 1$ in the identity above, which reduces it exactly to $\frac{N-1}{N}\sigma^2$. Comparing PROSE's $\frac{M-1}{M}(1-\rho_r)\sigma^2$ at $M = 4$ against that cross-design baseline gives

$$R_{\mathrm{dep}} = \frac{(M-1)(1-\rho_r)N}{M(N-1)} = \frac{3(0.28)(16)}{4(15)} = 0.224, \qquad 1 - R_{\mathrm{dep}} = 0.776.$$

This is a *cross-design* comparison: it contrasts the deployed i.i.d. baseline with the clustered hierarchical one, and is the right number when asking what PROSE buys over GRPO as shipped. It is not the number a same-batch recomputation would return.

Protocol-matched validation. The advantage-variance ratio measured by recomputing both estimators on the stored k=4, M=4, N=16 batch is 0.26 (Appendix K.4), against the protocol-matched same-batch prediction $R_{\mathrm{same}} = 0.262$. This is the direct quantitative test of Theorem 4.1(a)/(b); the deployed instantiation answers the different, cross-design question above and is reported separately so the two are never conflated. As a safeguard against fitting and predicting on the same data, we additionally specify — but have not yet run — a split-half check: ρ_r fitted on one half of the problems and used to predict the variance ratio on the held-out half (Appendix K.4).

Asymptotics, in the direction that argues against us. For both instantiations, holding M fixed and letting the number of plans grow,

$$1 - R \;\longrightarrow\; 1 - \frac{(M-1)(1-\rho_r)}{M} \;=\; \frac{1 + (M-1)\rho_r}{M} \;\geq\; \rho_r,$$

with equality only at $\rho_r = 1$; at M=4, ρ_r=0.72 this fixed-M limit is 0.79. Recovering the clean asymptotic value ρ_r requires $M \to \infty$ as well. The finite-M regime we operate in therefore overstates the asymptotic ρ_r rather than understating it, and we report the fixed-M limit rather than the more flattering ρ_r headline.

J Iteration-Complexity Corollary and Proof

Theorem 4.1(c) is a per-step statement about the policy-improvement ratio. Under a standard convergence regime it lifts to a statement about the number of iterations each estimator needs. We restate the assumptions it inherits, proved and discussed in Appendix H:

- **(A1)** cross-plan score orthogonality (exact under the idealised model; approximate in practice because of the shared problem-description prefix, with measured $\widehat{\rho}_{\text{cross}} \approx 0.08$);
- **(A2)** non-negative within-plan score correlation, $\rho_h^{\text{flat}} \geq 0$;
- **(A3)** the score-norm condition $\|h^{\text{cond}}\|^2 \geq (M-1)(1-\rho_h^{\text{flat}})\|h^{\text{plan}}\|^2$ (sufficient; the exact requirement carries an extra factor $1+(M-1)\rho_h^{\text{flat}}$ on the left).

Corollary J.1 (Iteration-complexity ratio)**.** *Assume J is L-smooth and μ-PL (Karimi et al., 2016), assumptions (A1)–(A3) of Theorem 4.1(c), and let $\gamma^{\text{flat}}, \gamma^{\text{cond}}, \gamma^{\text{plan}}$ denote the trajectory-uniform noise-to-signal bounds $\gamma^{\bullet} := \|h^{\bullet}\|^2\sigma^2/(N\|\nabla J(\theta_t)\|^2)$, with $\gamma^{\text{flat}} = \gamma^{\text{cond}} + \gamma^{\text{plan}}$. Then the PL-derived sufficient iteration counts to reach ε-accuracy satisfy*

$$\frac{T_{\text{suff}}^{\text{PROSE}}(\varepsilon)}{T_{\text{suff}}^{\text{GRPO}}(\varepsilon)} \;\leq\; \frac{1+\gamma^{\text{cond}}(1-\rho_r)+\gamma^{\text{plan}}\big(1+(M-1)\rho_r\big)}{1+\gamma^{\text{flat}}\big(1+(M-1)\rho_h^{\text{flat}}\rho_r\big)} \;\leq\; 1, \tag{83}$$

the second inequality strict when $\rho_r > 0$ and (A3) holds strictly.

Proof. Write $S^{\text{GRPO}} := \gamma^{\text{flat}} D_{\text{eff}}$ with $D_{\text{eff}} = 1+(M-1)\rho_h^{\text{flat}}\rho_r$, and $S^{\text{PROSE}} := \gamma^{\text{cond}}(1-\rho_r)+\gamma^{\text{plan}}(1+(M-1)\rho_r)$; these are exactly the gradient-variance-to-signal bounds $\Sigma_t^{\bullet}/\|\nabla J(\theta_t)\|^2$ established in Appendix H. From the descent lemma for L-smooth objectives with the step size optimised separately for each estimator, each estimator's per-step progress is at least $\|\nabla J(\theta_t)\|^2/\big(2L(1+S)\big)$ with its own S. Applying the PL inequality $\|\nabla J\|^2 \geq 2\mu(J^*-J)$ and iterating yields the geometric-rate bound $\mathbb{E}[J^*-J(\theta_T)] \leq \big(1-\mu/(L(1+S))\big)^T (J^*-J(\theta_0))$, whose sufficient iteration count is

$$T_{\text{suff}}(\varepsilon) = \frac{L\,(1+S)\log(1/\varepsilon)}{\mu}, \tag{84}$$

where S is the corresponding noise-to-signal bound (S^{GRPO} or S^{PROSE} above). Taking the ratio gives the first inequality in Equation (83).

For the second inequality, compare numerator and denominator of Equation (83) termwise, substituting $\gamma^{\text{flat}} = \gamma^{\text{cond}} + \gamma^{\text{plan}}$ and collecting the γ^{cond} and γ^{plan} coefficients separately. This is the same termwise comparison that establishes Theorem 4.1(c) in Appendix H, and it yields exactly

$$\big[\text{numerator}\big] - \big[\text{denominator}\big] \;=\; -\rho_r\Big\{\gamma^{\text{cond}}\big[1+(M-1)\rho_h^{\text{flat}}\big] - \gamma^{\text{plan}}(M-1)\big(1-\rho_h^{\text{flat}}\big)\Big\} \;\leq\; 0. \tag{85}$$

Indeed, the γ^{cond} coefficient is $(1-\rho_r)-1-(M-1)\rho_h^{\text{flat}}\rho_r = -\rho_r[1+(M-1)\rho_h^{\text{flat}}]$ and the γ^{plan} coefficient is $(1+(M-1)\rho_r)-1-(M-1)\rho_h^{\text{flat}}\rho_r = (M-1)\rho_r(1-\rho_h^{\text{flat}})$. The braced quantity in Equation (85) is non-negative precisely when $\gamma^{\text{cond}}[1+(M-1)\rho_h^{\text{flat}}] \geq \gamma^{\text{plan}}(M-1)(1-\rho_h^{\text{flat}})$, which, since $\gamma^{\bullet} \propto \|h^{\bullet}\|^2$, is the exact form of (A3); the stated sufficient form (A3) implies it because $1+(M-1)\rho_h^{\text{flat}} \geq 1$ under (A2). Finally, T_{suff} in Equation (84) is strictly increasing in S, so the inequality between the two $1+S$ terms transfers directly to the iteration counts. Strictness holds whenever $\rho_r > 0$ and (A3) holds strictly with $\gamma^{\text{cond}} > 0$. □

Remark J.2 (Why the individual γ's must be retained). It is tempting to simplify Equation (83) by replacing $\gamma^{\text{cond}}, \gamma^{\text{plan}}$ with a common upper bound $\bar{\gamma} \geq \max(\gamma^{\text{cond}}, \gamma^{\text{plan}}, \gamma^{\text{flat}})$, which would bound the numerator by $1+\bar{\gamma}(1+(M-1)\rho_r)$ and leave the comparison $1+(M-1)\rho_r$ versus $D_{\text{eff}} = 1+(M-1)\rho_h^{\text{flat}}\rho_r$. That comparison requires $\rho_h^{\text{flat}} \geq 1$, which for a correlation coefficient holds only in the degenerate perfectly-collinear case and is excluded by the measured $\widehat{\rho}_h^{\text{flat}} \approx 0.56$ (Appendix L). We therefore do not use that relaxation anywhere: the coarse bound is genuinely unavailable, and the termwise comparison Equation (85), which keeps γ^{cond} and γ^{plan} separate and exploits the fact that the plan-level score carries a small fraction of the total score norm, is what delivers the result. The same caution applies to substituting $\gamma^{\text{cond}} \leq \gamma^{\text{flat}}$ and $\gamma^{\text{plan}} \leq \gamma^{\text{flat}}$ simultaneously: those two substitutions move the numerator in the same direction as the denominator and do not by themselves yield Equation (83).

Remark J.3 (Which GRPO the corollary compares against). As in Appendix I.1, the denominator of Equation (83) is the *counterfactually clustered* GRPO gradient: D_{eff} is a same-batch quantity, formed by asking what GRPO's flat baseline would cost on a batch that happens to contain k plans of M rollouts. The *deployed* analogue sets $D_{\text{eff}} \to 1$ (i.i.d. sampling, no within-batch plan structure), and repeating the termwise step with $D_{\text{eff}} = 1$ gives $[\text{num}] - [\text{den}] = \rho_r\big[(M{-}1)\gamma^{\text{plan}} - \gamma^{\text{cond}}\big]$, so the requirement becomes the threshold $\|h^{\text{cond}}\|^2 \geq (M{-}1)\|h^{\text{plan}}\|^2$, i.e. a norm ratio of 3 at $M{=}4$. This coincides with the exact form of (A3) evaluated at $\rho_h^{\text{flat}} = 0$, and is the more demanding of the two thresholds. Both are cleared by the token-count proxy margin quantified in Theorem J.4, which also records the second reading of that ratio, under which the more demanding one is not.

Remark J.4 (How much margin (A3) has at the operating point). At $M{=}4$ and $\widehat{\rho}_h^{\text{flat}} \approx 0.56$ (Appendix L), the exact-form threshold on the score-norm ratio $\|h^{\text{cond}}\|^2/\|h^{\text{plan}}\|^2$ is $(M{-}1)(1 - \rho_h^{\text{flat}})/[1 + (M{-}1)\rho_h^{\text{flat}}] = 3(0.44)/2.68 \approx 0.49$, and the stated sufficient form requires $3(0.44) = 1.32$. Under the paper's norm heuristic, read as proportional to token count (~ 35 plan tokens against $\sim 200{+}$ code tokens per rollout), the realised ratio is ≈ 5.7: both forms hold with at least a $4\times$ margin, and the exact form survives even in the worst admissible case $\rho_h^{\text{flat}} = 0$, where the threshold rises to 3. That margin is specific to the token-count reading of the ratio, and we flag that it is not the only one available. The Pythagorean identity Equation (59), applied to the measured $\widehat{\rho}_h^{\text{flat}} = 0.56$ and $\widehat{\rho}_h^{\text{cond}} = 0.21$, gives a ratio of ≈ 1.26 instead (Equation (72)); that value clears the exact-form threshold 0.49 by $2.6\times$ but clears neither the sufficient form 1.32 nor the worst case 3. The two proxies are not calibrated against each other and we do not reconcile them (Appendix H.7). (A3) is therefore an explicit, checkable condition rather than a hidden one, and the exact form — the one this corollary's termwise step actually uses, against the counterfactually clustered comparator — is not close to binding under either reading. Substituting $\gamma^{\text{cond}}/\gamma^{\text{plan}} \approx 5.7$, $\rho_r = 0.72$ and $\widehat{\rho}_h^{\text{flat}} = 0.56$ into Equation (85) gives a margin of $\approx -10.0\,\gamma^{\text{plan}}$; at the ratio 1.26 it is $\approx -1.5\,\gamma^{\text{plan}}$, still negative. We do not convert this into a claimed training-time speed-up: Equation (83) bounds a ratio of PL-derived *sufficient* iteration counts, the norm ratio is a token-count proxy rather than a measurement, and ρ_h^{flat} itself is available only as a cosine proxy. The quantitative test of the theory in this paper is the advantage-variance ratio (0.262 predicted against 0.26 measured, Appendix I.1), not any iteration-count translation.

Remark J.5 (Scope of the corollary). Three limitations are worth making explicit. First, Equation (83) compares PL-derived sufficient iteration counts, i.e. upper bounds, not the unknown true complexities of the two procedures; a ratio of upper bounds below 1 certifies that the PROSE bound is the tighter of the two, which is weaker than certifying faster convergence. Second, the L-smooth/μ-PL regime is an idealisation for policy-gradient training on a non-stationary objective; we use it, as is standard, to expose the dependence on the noise-to-signal ratio rather than to predict wall-clock training time, which we measure directly in the compute accounting of Section 5. Third, the corollary inherits (A1)–(A3) in full, including the approximate nature of (A1): the corollary degrades continuously in ρ_{cross} by the same first-order analysis given in Appendix H, and at the measured $\widehat{\rho}_{\text{cross}} \approx 0.08$ the correction does not change the sign of Equation (85). The corollary is best read, like Theorem 4.1(c), as design rationale: it identifies ρ_r and the score-norm split as the quantities that govern the comparison, and says the comparison is favourable at the regime we actually train in.

K Empirical Measurement of the Latent-Plan Reward Correlation

The entire argument of Section 3.2 and Theorem 4.1 rests on a single empirical premise: that execution rewards are not homogeneous across the N rollouts of a group, but follow a latent-plan mixture in which two rollouts realizing the same plan share a substantial fraction of their reward variance. This appendix measures that fraction. Appendix K.1 gives the pre-training measurement, Appendix K.2 states precisely what the measured quantity does and does not assert about GRPO, Appendix K.3 documents the retrospective plan-extraction pipeline used to track the correlation through training on rollouts that carry no plan text, and Appendix K.4 documents the protocol behind the paper's one direct, mapping-free test of Theorem 4.1(a): the advantage-variance ratio.

K.1 Pre-training measurement protocol

We measure the intraclass correlation coefficient (ICC) of execution rewards within plan clusters on a held-out subset of APPS+ problems using the base Qwen2.5-Coder-7B model for APPS+ and Qwen2.5-Instruct-7B for GSM8K, both *before any RL training*, so that the measurement characterizes the pre-training generative distribution rather than a property induced by our own objective. For each of 500 randomly sampled problems we generate $k{=}4$ plans and 8 implementations per plan (oversampled beyond the training operating point of $M{=}4$ purely to reduce estimation noise in $\widehat{\rho}_r$), execute all programs against the problem's test suite, and compute the one-way random-effects ICC per problem from the between- and within-plan mean squares (Scheffé, 1959; Searle et al., 2006),

$$\widehat{\rho}_r \;=\; \frac{\mathrm{MS}_B - \mathrm{MS}_W}{\mathrm{MS}_B + (M_{\mathrm{meas}} - 1)\,\mathrm{MS}_W}, \qquad M_{\mathrm{meas}} = 8, \tag{86}$$

truncated at zero, and average across problems. The reported $\pm$ is the standard deviation of the per-problem ICC across the 500 problems, not a standard error: it describes genuine heterogeneity in how strongly a problem's outcome is determined by strategy rather than by implementation detail, which is itself the quantity that the diversity- and difficulty-conditioned analyses of Appendix O resolve. The one-way random-effects decomposition $\sigma^2_B := \mathrm{Var}(v_i)$, $\sigma^2_W := \mathbb{E}[\mathrm{Var}(r \mid v_i)]$, $\sigma^2 = \sigma^2_B + \sigma^2_W$ covers Bernoulli rewards without modification ($\sigma^2_W = \mathbb{E}[v_i(1 - v_i)]$, with v_i the plan-conditional success probability), which is the case relevant here.

Table 11: Empirical intraclass correlation (ICC) of execution rewards within plan clusters, measured on 500 problems per domain before RL training. The design-effect rows are a *counterfactual* clustered-sampling translation of ρ_r into effective-sample-size units at the training operating point $M{=}N/k{=}4$; they are not a claim that GRPO draws rollouts in clusters (see Appendix K.2).

Metric	**APPS+ (code)**	**GSM8K (math)**
Mean ICC $\widehat{\rho}_r$ (across problems)	0.72 ± 0.14	0.48 ± 0.16
Median ICC	0.78	0.51
% of problems with ICC > 0.5	81.4%	57.0%
Counterfactual design effect $D_r = 1{+}(M{-}1)\rho_r$	3.16	2.44
Counterfactual N_{eff}/N	0.32	0.41

The mean ICC of 0.72 on APPS+ establishes strong within-plan reward correlation in code generation: programs sharing a plan succeed or fail together far more often than chance, and on 81.4% of problems a majority of the reward variance is between plans rather than within them. The lower but still substantial ICC of 0.48 on GSM8K shows that the same latent-plan structure is present in mathematical reasoning with lower severity (solution strategies are somewhat less deterministic than code algorithms), yielding a smaller counterfactual design effect ($D_{\mathrm{math}}{=}2.44$ versus $D_{\mathrm{code}}{=}3.16$) and, consistently, a smaller though still positive gain from PROSE (Table 27). The ordering $D_{\mathrm{math}} < D_{\mathrm{code}}$ predicting a smaller gain is the only cross-domain test the data support: the gradient-level quantity ρ_h^{flat} was *not* measured on GSM8K, so no quantitative cross-domain prediction is made.

K.2 What the measured ICC does and does not assert

Because this quantity carries the paper's diagnosis, it is worth stating its scope precisely.

It does assert that the reward distribution over a group of rollouts for a fixed problem is a latent-plan mixture with a non-degenerate between-plan component: writing $r_{i,j} = \mu + \alpha_i + e_{i,j}$ with $\mathrm{Var}(\alpha_i) = \sigma^2_B$ and $\mathrm{Var}(e_{i,j}) = \sigma^2_W$, the measurement says $\sigma^2_B/\sigma^2 \approx 0.72$ on APPS+. Under this mixture the flat group-relative baseline, the minimum-variance choice in the homogeneous case $\sigma^2_B = 0$, is variance-suboptimal, and the efficient baseline is the plan-conditioned one (Theorem 4.1(b)).

It does not assert that GRPO's sampling is defective. Under GRPO's i.i.d. draws from the marginal $\pi_\theta(y \mid x) = \sum_p \pi_\theta(p \mid x)\pi_\theta(y \mid x, p)$ the N rollouts are marginally exchangeable, the unconditional covariance between two draws is zero, and the grand mean has variance σ^2/N exactly. Nothing is silently lost at the sampling stage. The design-effect rows of Table 11 are a unit-bearing restatement of ρ_r — what the effective sample size of a group of N would be if the same rollouts were,

counterfactually, organized as k plan clusters of size M — offered because effective-sample-size units are easier to reason about than a raw correlation. The substantive inefficiency is at the *advantage* level, where the same N rollouts serve as both target and baseline, and it is quantified by Theorem 4.1(a) rather than by Equation (2). Appendix K.4 tests exactly that advantage-level statement.

K.3 Retrospective plan extraction: tracking the correlation through training

A measurement taken before RL leaves open the possibility that the latent-plan structure is an artifact of the pre-trained model that RL quickly destroys. Figure 2 therefore tracks the ICC at checkpoints throughout training, for both PROSE and GRPO.

For PROSE this is direct: the policy emits an explicit plan, so cluster membership is observed and the ICC is computed on the realized $k{=}4$ plan groups with the same estimator as Equation (86). The ICC rises slightly during PVE warm-up, as the model learns to commit more consistently to its plans, and stabilizes at ≈ 0.77 throughout APRA training, confirming that plan structure remains informative across the full run rather than only at initialization.

For GRPO no such grouping variable exists. **Unconditioned GRPO emits no plan text**, so there is nothing in its rollouts to cluster directly; the grouping variable must instead be recovered from the generated code. We do so with an explicitly retrospective, read-only pipeline, which is also the pipeline reused by the post-hoc control of Appendix M and stress-tested in Appendix N:

1. **Extraction.** For each stored rollout of a GRPO checkpoint — a complete generated program, produced with no plan prompting of any kind — an auxiliary instruction-tuned LLM (Qwen2.5-Instruct-7B, the same family as the policy, used only for extraction, never updated, and never run inside any training loop) is prompted with a fixed template to state, in the format of the plan-generation prompt of Appendix S, the algorithmic strategy that the program implements. The extractor sees only the problem statement and the program, never the execution outcome, so the recovered label cannot be contaminated by the reward it is later used to cluster.
2. **Embedding.** Each extracted plan text is embedded with the policy's final-layer hidden state at the `</plan>` token, the same representation used for forward plan embeddings elsewhere in the paper.
3. **Clustering.** The embeddings for a problem are partitioned into $k{=}4$ groups by greedy cosine assignment to centroids, yielding the retrospective cluster labels.
4. **ICC.** The already-stored execution rewards are then decomposed over these retrospective clusters with the estimator of Equation (86).

Two properties of this pipeline matter for how Figure 2 should be read. First, it is purely observational: GRPO's training is untouched, no rollout is regenerated, and no plan is ever supplied to the policy; the procedure only relabels rollouts that GRPO produced and trained on. Second, because the grouping variable is estimated rather than observed, the extractor, the cluster count and the embedding layer are free parameters of the measurement, and a poor choice would attenuate the ICC toward the random-grouping floor. Appendix N varies all three.

The retrospective ICC for GRPO decays from its pre-training level toward ≈ 0.40 by the end of training: latent plan structure remains clearly present in unconditioned rollouts, but the alignment between rollouts weakens as the policy sharpens. Two anchors bracket the number and make it interpretable. Random grouping of the same rollouts gives ICC ≈ 0.05, establishing that the pipeline returns near-zero when the grouping carries no semantic content, so the measured 0.40 is not a clustering artifact. Forward-sampled PROSE, where cluster membership is observed rather than estimated, gives ≈ 0.77. The retrospective GRPO value sits roughly eight times the floor and well below the observed-membership ceiling, which is what one expects from a noisy estimate of a real grouping variable.

Taken together, the pre-training measurement and the training curves support the modelling assumption in the form the theory needs it: the latent-plan reward correlation is large at initialization, persists under plan-conditioned training, and — crucially for the claim that PROSE exploits pre-existing structure rather than manufacturing it — remains substantial in rollouts generated with no plan conditioning at all. Appendix M converts that observational statement into an interventional one.

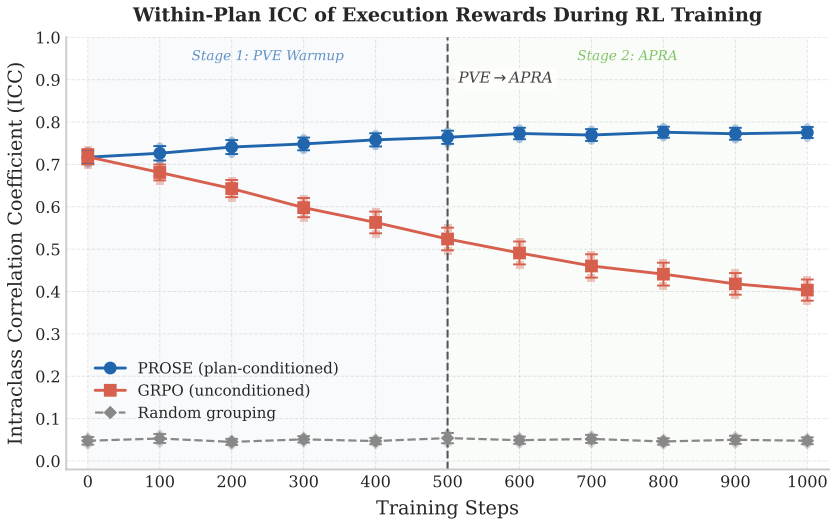

Figure 2: Within-plan ICC of execution rewards measured at training checkpoints. For PROSE, cluster membership is *observed*: the policy emits an explicit plan and the ICC is computed on the realized k=4 plan groups, which stay informative (≈ 0.77) across the full run. For GRPO, which emits no plan text, cluster membership is *recovered retrospectively*: an auxiliary instruction-tuned LLM extracts a short algorithmic plan from each already-generated program, the extracted plans are embedded at the `</plan>` token and greedily clustered by cosine into k=4 groups, and the stored execution rewards are decomposed over those clusters (Appendix K.3). This is a read-only analysis of unmodified GRPO rollouts, not a variant of GRPO. The random-grouping curve applies the identical decomposition to randomly permuted group labels and is the floor of the measurement; absolute levels of the retrospective curve are extractor-dependent within the range reported in Appendix N, and the claim rests on the decay shape rather than the endpoint value. The dashed vertical line marks the PVE→APRA transition at step 500.

K.4 Direct test of the advantage-variance identity

Theorem 4.1(a) makes a prediction that requires no mapping from variance to task performance and no free parameters: the ratio of the plan-conditioned advantage variance to the flat advantage variance, at a given batch design and a given ρ_r. This is the paper's most direct quantitative test of the theory, and it is a *variance* test, not a pass@1 test; the budget-curve translation used elsewhere in the evaluation is a secondary sanity check and is labelled as such.

Recomputation protocol. Both estimators are recomputed offline, on one stored clustered batch, at the training operating point:

- **Batch design:** k=4 plans $\times$ M=4 rollouts per plan, N=16, over the 500 held-out problems. This is the design PROSE actually trains with. It is deliberately not the oversampled M=8 design of Appendix K.1, which exists only to estimate ρ_r precisely.
- **Shared rewards:** the two estimators are applied to the identical stored reward vector; only the baseline subtracted differs (grand mean over all N versus the plan-conditional mean over the M rollouts sharing a plan). No rollout is regenerated, so the comparison is exactly the variance of two estimators on one sample, with sampling noise held fixed.
- **Aggregation:** the advantage variance is computed per problem over its N rollouts and averaged across the 500 problems, for both estimators, before the ratio is formed.
- **Parameter provenance:** ρ_r=0.72 entering the prediction is the pre-training measurement of Table 11, taken on a different batch design and never fitted to the variance ratio.

Protocol-matched prediction. The prediction is the *same-batch instantiation* of the one-way ANOVA identity of Appendix G, evaluated at the same (k, M) as the recomputation. Writing

$\mathrm{Var}(\widehat{A}^{\mathrm{flat}}_{i,j}) = \frac{k-1}{k}\sigma_B^2 + \frac{N-1}{N}\sigma_W^2$ and $\mathrm{Var}(\widehat{A}_{\mathrm{code},i,j}) = \frac{M-1}{M}\sigma_W^2 = \frac{M-1}{M}(1-\rho_r)\sigma^2$, the ratio is

$$R_{\mathrm{same}} \;=\; \frac{\frac{M-1}{M}(1-\rho_r)}{\frac{k-1}{k}\rho_r + \frac{N-1}{N}(1-\rho_r)}, \tag{87}$$

which at $k{=}M{=}4$, $N{=}16$, $\rho_r{=}0.72$ evaluates to $0.21\sigma^2/0.8025\sigma^2 = \mathbf{0.262}$. The measured ratio is $\mathbf{0.26}$. The corresponding reduction fraction is $1 - R_{\mathrm{same}} = 0.738$.

Why the design must be stated. The identity's prediction is a function of (k, M, N), so a prediction and a measurement are comparable only when both are evaluated at the same design. We therefore report protocol-matched pairs only. The other labelled instantiation of the same identity — the *deployed* comparator, in which GRPO's i.i.d. sampling is read as $k{=}N$, $M{=}1$ *in the flat term only* (each rollout carrying its own latent plan), so that the flat variance reduces exactly to $\frac{N-1}{N}\sigma^2 = 0.9375\sigma^2$ while the plan-conditioned numerator stays at PROSE's $M{=}4$, and the ratio becomes $R_{\mathrm{dep}} = (M-1)(1-\rho_r)N/(M(N-1)) = \mathbf{0.224}$ (reduction fraction 0.776) — answers a different, cross-design question: how a hierarchical baseline compares against GRPO as it is actually run, at equal N. It is the right figure for the deployment comparison and the wrong figure for this recomputation, and we label the two comparators wherever they appear (Theorem 4.1(a), Theorem J.1, Appendix H). Asymptotically the distinction vanishes: with M fixed and the number of plans growing, both reduction fractions converge to $[1 + (M{-}1)\rho_r]/M = 0.79$ at $M{=}4$, $\rho_r{=}0.72$, and the limit ρ_r itself requires $M \to \infty$ as well; the fixed-M scoping stated in the remark following Theorem 4.1 is retained unchanged.

Split-half safeguard. Since ρ_r and the measured ratio are both computed on the same 500 held-out problems, the check reported above is in-sample in its parameter. We therefore specify an out-of-sample version of it: $\widehat{\rho}_r$ is estimated on a random half of the problems, the prediction Equation (87) is formed from that estimate alone, and that prediction is compared against the measured ratio on the disjoint held-out half, so that the data used to fit the predicting parameter and the data used to evaluate the prediction never overlap. The protocol is fixed in advance, so the single comparison it licenses is the one that would be reported. That comparison has not been run: no split-half result appears in this paper, and the 0.262 versus 0.26 agreement above is the in-sample comparison.

Scope of the agreement. The match (0.262 predicted, 0.26 measured) is a test of the variance identity at the operating point, with ρ_r supplied externally. It does not by itself establish that the variance reduction is what produces the observed pass@1 gain; that link runs through Theorem 4.1(c) and is separately supported, with weaker evidence, by the budget-curve translation and by the post-hoc control of Appendix M.

L Empirical Measurement of Within-Plan Score Correlation

Theorem 4.1 depends not only on the reward ICC ρ_r but also on the within-plan score correlation ρ_h^{flat} (Equation (56)), which is what couples reward-level structure to gradient-level structure. Directly computing $\nabla_\theta \log \pi_\theta(y_{i,j} \mid x)$ for every rollout is prohibitive at 7B scale, so we use a tractable heuristic proxy: the cosine similarity between per-rollout last-layer hidden-state sequences, averaged over the generation span and normalized to unit norm. We do not claim that this hidden-state similarity equals the true score-function correlation; it is a representational similarity used as a directional indicator of within-plan score alignment. The numbers below should accordingly be read as order-of-magnitude estimates of ρ_h, not as exact measurements; this is why the paper declines to convert D_{eff} into a single point-estimate performance multiplier.

L.1 Estimators and measured values

Using the same 500-problem held-out APPS+ subset as Appendix K, with $k{=}4$ plans and $M{=}4$ rollouts per plan on Qwen2.5-Coder-7B (pre-RL), we compute

$$\widehat{\rho}_h^{\mathrm{flat}} \;=\; \frac{1}{kM(M-1)} \sum_{i,\, j \neq j'} \frac{\langle \bar{h}_{i,j}, \bar{h}_{i,j'} \rangle}{\|\bar{h}_{i,j}\| \, \|\bar{h}_{i,j'}\|}, \qquad \widehat{\rho}_h^{\mathrm{cond}} \;=\; \text{analogous, with plan-conditioned scores,}$$

and an analogous cross-plan estimator $\widehat{\rho}_{\text{cross}}$ over rollout pairs $(i,j),(i',j')$ with $i \neq i'$. The measured values are

$$\widehat{\rho}_h^{\text{flat}} = 0.56 \pm 0.09, \qquad \widehat{\rho}_h^{\text{cond}} = 0.21 \pm 0.06, \qquad \widehat{\rho}_{\text{cross}} = 0.08 \pm 0.03,$$

with mean and standard deviation taken across the 500 problems; Table 12 collects them.

Table 12: Hidden-state cosine proxies for within- and cross-plan score correlation on the APPS+ holdout (k=4, M=4, Qwen2.5-Coder-7B pre-RL, 500 problems; mean ± std across problems). Read as order-of-magnitude estimates rather than exact measurements of ρ_h.

Quantity	Estimate	Role
$\widehat{\rho}_h^{\text{flat}}$ (within-plan, unconditioned)	0.56 ± 0.09	enters D_{eff}; assumption (A2)
$\widehat{\rho}_h^{\text{cond}}$ (within-plan, plan-conditioned)	0.21 ± 0.06	residual after removing h^{plan}
$\widehat{\rho}_{\text{cross}}$ (cross-plan, shared problem prefix)	0.08 ± 0.03	(A1) cross-plan orthogonality

The ordering $\widehat{\rho}_h^{\text{flat}} > \widehat{\rho}_h^{\text{cond}} > \widehat{\rho}_{\text{cross}}$ is consistent with the structural decomposition $h^{\text{flat}} = h^{\text{plan}} + h^{\text{cond}}$: conditioning on the plan removes the shared h^{plan} component, cutting the within-plan similarity by roughly a factor of 2.7, and the residual cross-plan similarity attributable to the shared problem prefix is small. The last row is the empirical status of assumption (A1): cross-plan score orthogonality holds approximately rather than exactly, at $\widehat{\rho}_{\text{cross}} \approx 0.08$, small enough that even a sign-flipped worst-case reading does not dominate the within-plan terms (Appendix H). The middle row is the empirical status of assumption (A2), $\rho_h^{\text{flat}} \geq 0$, which is satisfied with a wide margin.

L.2 Consequences for the gradient-level statements

Gradient design effect. Combining $\widehat{\rho}_h^{\text{flat}} = 0.56$ with $\rho_r = 0.72$ at the operating point M=4 gives the within-method gradient design effect of Theorem 4.1(c),

$$D_{\text{eff}} \;=\; 1 + (M-1)\,\widehat{\rho}_h^{\text{flat}}\,\rho_r \;=\; 1 + 3(0.56)(0.72) \;\approx\; 2.21,$$

against the counterfactually clustered comparator (the same-batch comparator of Appendix K.4; the deployed analogue has $D_{\text{eff}} \to 1$). Because $\widehat{\rho}_h^{\text{flat}}$ is a proxy, its ± 0.09 dispersion propagates directly into D_{eff}, and we therefore use D_{eff} as design rationale and as the input to a labelled sanity check, never as a headline quantitative prediction. The direct quantitative test of the theory remains the advantage-variance ratio of Appendix K.4, which involves ρ_h^{flat} not at all.

Empirical status of assumption (A3). Theorem 4.1(c) requires, in its sufficient form, $\|h^{\text{cond}}\|^2 \geq (M-1)(1-\rho_h^{\text{flat}})\|h^{\text{plan}}\|^2$; the exact requirement carries an additional factor $1 + (M-1)\rho_h^{\text{flat}}$ on the left-hand side. At $\widehat{\rho}_h^{\text{flat}} = 0.56$ and M=4 these translate into score-norm ratio thresholds of

$$\frac{\|h^{\text{cond}}\|^2}{\|h^{\text{plan}}\|^2} \;\geq\; (M-1)(1-\rho_h^{\text{flat}}) = 1.32 \qquad \text{(sufficient form)},$$

$$\frac{\|h^{\text{cond}}\|^2}{\|h^{\text{plan}}\|^2} \;\geq\; \frac{1.32}{1+(M-1)\rho_h^{\text{flat}}} = \frac{1.32}{2.68} \approx 0.49 \qquad \text{(exact form)}.$$

The paper's norm heuristic reads the score norm as growing with the number of generated tokens the component is responsible for; under that reading, with roughly 35 plan tokens against 200+ code tokens per problem, the ratio $\kappa := \|h^{\text{cond}}\|^2/\|h^{\text{plan}}\|^2$ is ≈ 5.7, which clears both thresholds and also the degenerate-case threshold $(M-1) = 3$ that the deployed comparator requires. That reading is not the only one our own numbers support, and the two do not agree. The Pythagorean identity of Theorem H.2 forces $(1+\kappa)\rho_h^{\text{flat}} = 1 + \kappa\,\rho_h^{\text{cond}}$, so the proxies of Table 12 imply $\kappa = (1-\rho_h^{\text{flat}})/(\rho_h^{\text{flat}} - \rho_h^{\text{cond}}) = 0.44/0.35 \approx 1.3$ — a plan prefix carrying $\approx$44% of the squared score norm rather than the $\approx$15% the token count assumes. A hidden-state cosine and a token count measure different objects and are not calibrated against each other, and we do not reconcile them; we state instead what survives the less favourable reading. At $\kappa \approx 1.3$ the exact form above still holds, while the sufficient form (1.32) and the deployed-comparator threshold (3) do not. What matters is that (A3) was previously implicit: it is a real condition, now stated, and at our operating point it is satisfied in its exact form against the counterfactually clustered comparator that Theorem 4.1(c) is stated for. A direct measurement of κ would settle the disagreement and would be a cleaner check than either proxy; Appendix H.7 carries the same discussion at the point where the condition is used.

What is not measured. ρ_h^{flat} was measured only on APPS+ with Qwen2.5-Coder-7B. It was not measured on GSM8K, so the mathematics results support only the ordering test of Appendix K.1 ($D_{\text{math}} < D_{\text{code}}$ predicts a smaller gain), not a quantitative cross-domain prediction. Nor was it re-measured at training checkpoints; the training-time persistence evidence in this paper is at the reward level (Figure 2), not the gradient level.

M Post-hoc Plan Extraction Control

M.1 The question this control answers

PROSE changes two things at once relative to GRPO+CoT. It changes the *generative process*, in that rollouts are conditioned on one of k explicitly sampled shared plans, and it changes the *estimator*, in that advantages are taken relative to a plan-conditional baseline rather than a flat group mean. Only the second is what Theorem 4.1 is about. A reader is entitled to ask whether the measured gain is a statistical effect of better credit assignment on structure that was already there, or whether plan conditioning simply turns the task into an easier one, in which case the theory would be decoration on an unrelated empirical finding.

The evidence in Appendix K is observational: the retrospective ICC shows that substantial latent-plan structure exists in rollouts generated with no plan conditioning at all. This appendix makes the corresponding interventional statement: hierarchical credit assignment on retrospectively recovered clusters is by itself beneficial, with the generative process held fixed.

M.2 Protocol

The PVE-posthoc run holds generation identical to GRPO+CoT and changes only the estimator:

1. **Generation.** 16 independent CoT rollouts per problem, exactly as in GRPO+CoT. There is no forward plan conditioning beyond the model's own CoT scratchpad: no plan is sampled in advance, no plan is shared across rollouts, and no plan text is injected into any prompt. Every rollout is an i.i.d. draw from the same marginal that GRPO+CoT samples.
2. **Post-hoc extraction.** After generation, each rollout's plan is recovered by the auxiliary-LLM pipeline of Appendix K.3: the auxiliary model states the algorithmic strategy the program implements, without access to the execution outcome.
3. **Clustering.** The extracted plans are embedded at the `</plan>` token and greedily clustered by cosine into k=4 groups per problem.
4. **Objective.** Hierarchical advantages are applied to the extracted clusters under the PVE objective with ω=1. Adaptive allocation is off — APRA cannot be run here, since there is no forward plan to allocate budget to — so this run isolates the hierarchical-advantage component of the method.
5. **Training and evaluation.** 1000 steps, all other hyperparameters identical to the GRPO+CoT baseline (Appendix R); evaluation on LiveCodeBench under the standard protocol.

Because retrospective assignment does not control cluster sizes, the realized cluster sizes M_i vary across a problem's k groups; the plan-conditional baseline uses the realized per-cluster counts, so the balanced-M form of Theorem 4.1(a) is an idealization of this run rather than an exact description of it. This is one of several reasons to expect PVE-posthoc to underperform forward PVE, and it makes the comparison conservative in the direction of the hypothesis being tested.

M.3 Result

PVE-posthoc reaches 19.7 LiveCodeBench pass@1, against GRPO+CoT at 18.4 and forward PVE at 20.3 (Table 13). Post-hoc hierarchical credit assignment alone therefore recovers +1.3 of the +1.9 CoT→PVE step (just over two thirds of it) with the generative process completely unchanged, leaving a residual gap of 0.6 to forward PVE.

Table 13: Post-hoc plan extraction control on LiveCodeBench pass@1 (N=16, Qwen2.5-Coder-Instruct-7B). PVE-posthoc holds generation identical to GRPO+CoT and changes only the advantage estimator, applying hierarchical advantages to clusters recovered after the fact. Baseline and forward-PVE figures are the 3-seed means of Table 2; no seed dispersion is reported for the post-hoc run.

Run	Plan source	LCB pass@1	Δ vs. GRPO+CoT
GRPO + CoT	none (flat baseline)	18.4 ± 0.4	—
PVE-posthoc	extracted after generation	19.7	$+1.3$
PVE (forward)	sampled before generation, shared	20.3 ± 0.3	$+1.9$

M.4 Interpretation

The $+1.3$ is the theory-covered component, isolated. In this run the policy samples from precisely the distribution GRPO+CoT samples from; the only difference is which baseline is subtracted. The improvement can therefore not be attributed to plan conditioning making generation easier, to a changed exploration distribution, or to the shared-prefix effect. It is attributable to the estimator, and the estimator is exactly what Theorem 4.1(a)–(b) analyses. The control therefore separates exploiting latent strategy structure that already exists in independently generated rollouts from plan conditioning changing the task. Note also that the $+1.3$ effect of explicit plan generation on its own is separately accounted for and credited outside the theory, as the GRPO→GRPO+CoT step (Table 3); the two are distinct mechanisms that happen to carry the same magnitude at this operating point, and neither is being counted twice.

The residual 0.6 quantifies generation–estimation coupling. Forward PVE retains an advantage of 0.6 over the post-hoc control, and the gap has a straightforward reading. When rollouts are generated from a shared, explicit plan prefix, cluster membership is observed rather than estimated, so there is no label noise; cluster sizes are balanced by construction; and within-plan coherence is higher, which is the same property that Figure 2 measures as the difference between the observed-membership ICC (≈ 0.77) and the retrospective one (≈ 0.40). The residual therefore measures the value of *coupling* generation to the estimator's grouping structure, not an additional statistical mechanism. The decomposition is honest in both directions: had the gap been zero, the claim would have been stronger still (a purely statistical effect); had it been $+1.9$, forward conditioning would have been shown to be necessary and the mechanism story would have had to be re-scoped. The measured split places most of the effect on the estimator side and a real but minority share on the coupling side.

Relation to the retrospective ICC. The two pieces of evidence are the observational and interventional halves of one claim. Figure 2 shows that the correlation pre-exists plan conditioning: unconditioned GRPO rollouts carry retrospectively recoverable plan structure at ICC ≈ 0.40, far above the ≈ 0.05 random-grouping floor. This control shows that exploiting that pre-existing correlation is itself beneficial, without changing what the policy samples. Neither alone would settle the question; together they place the mechanism where the theory puts it.

Limitations of this control. Three caveats. The retrospective labels are produced by a model from the same family as the policy, so the extraction quality is not independent of the policy's own representations (Appendix N addresses the analogous concern for the ICC measurement by swapping the extractor). Extraction adds an inference pass per rollout, so PVE-posthoc is a diagnostic rather than a practical recipe; the practical recipe remains forward PVE. And because APRA cannot be run without a forward plan, this control speaks only to the hierarchical-advantage component ($+1.9$ of the $+5.3$), not to the allocation component.

N ROBUSTNESS OF THE RETROSPECTIVE ICC MEASUREMENT

N.1 WHY ROBUSTNESS MATTERS HERE

The retrospective ICC of Figure 2 is the one measurement in the paper whose grouping variable is estimated rather than observed. Its value therefore depends on three choices that are properties of the measurement pipeline rather than of the data: which model extracts the plan, how many clusters the extracted plans are partitioned into, and which hidden representation is used to embed them. A result that moved substantially under any of these would not support a claim about GRPO's rollouts; it would be a claim about our extractor.

N.2 VARIANTS

We re-ran the pipeline of Appendix K.3 on the same stored GRPO checkpoint rollouts, so that no generation is repeated and all variation is attributable to the measurement, varying one factor at a time:

1. **Plan extractor.** The default extractor is an instruction-tuned model from the policy's own family. We repeated the extraction with an instruction-tuned model from an *unrelated* family, which removes the concern that the measured structure reflects a shared tokenizer, shared pre-training data, or shared representational idiosyncrasies between the extractor and the policy.
2. **Cluster count.** $k \in \{3, 4, 6\}$, spanning coarser and finer partitions than the $k{=}4$ used throughout the paper. Since the ICC estimator of Equation (86) depends on the cluster size through $(M_{\text{meas}} - 1)$, changing k changes both the partition and the estimator's normalization, making this the most demanding of the three perturbations.
3. **Embedding layer.** The plan embedding was taken from a mid-network layer instead of the final layer, testing whether the clustering depends on the specific representation used elsewhere in the paper.

N.3 RESULT AND HOW TO READ IT

End-of-training retrospective ICC lands in the range $\mathbf{0.36}$–$\mathbf{0.44}$ across all variants, with the same decay trend from the pre-training level over the course of training in every case. The default configuration's ≈ 0.40 sits inside this band rather than at its edge.

Two existing anchors bracket the procedure and fix the scale on which this band should be read. Random grouping of the same rollouts returns ICC ≈ 0.05: this is what the pipeline reports when the grouping variable carries no semantic information, and it is the floor. Forward-sampled PROSE, where membership is observed and no extraction is involved, returns ≈ 0.77: this is the ceiling the same estimator reaches when the grouping variable is exact. The retrospective GRPO value sits between them, roughly $8\times$ the floor and roughly half the ceiling, across every variant tested. The position between the anchors — not the specific endpoint — is what the measurement establishes.

The claim is carried by the decay shape, not the absolute level. We state this explicitly because it determines what the number may be used for. The absolute retrospective ICC is extractor-dependent within roughly ± 0.04 of 0.40, and it is an attenuated estimate of an unobserved grouping variable, so it must *not* be substituted into the design-effect identity as though it were a measured ρ_r for GRPO, nor compared numerically against the 0.72 of Table 11, which was measured with observed plan membership on a different model state. What the measurement does establish, robustly across extractor, cluster count and embedding layer, is (i) that unconditioned GRPO rollouts carry latent plan structure far above the random-grouping floor at every checkpoint, and (ii) that this structure attenuates monotonically over training. Both (i) and (ii) are qualitative facts about shape, and both are what the discussion around Figure 2 and the post-hoc control of Appendix M actually rely on.

Residual caveat. All variants share one design choice: plans are recovered by prompting a language model to describe a program's strategy, and clusters are formed by cosine geometry in a transformer hidden space. A qualitatively different notion of "same plan" — for example one grounded in

program analysis, in the set of test cases a program passes, or in human annotation — could produce a different partition, and we have not tested that. The robustness reported here is robustness within a family of pipelines, not across all reasonable definitions of a plan.

O Extended Evaluation and Ablations

This appendix collects the ablations and diagnostic analyses that support the main-text evaluation. Appendices O.1 to O.4 carry over the component, allocation-rule, noise-robustness and hyperparameter studies; Appendix O.5 audits the plan-diversity precondition that PROSE relies on; and Appendices O.6 to O.9 report four analyses that probe the mechanism directly: how the gain varies with the amount of genuinely distinct plan structure available, where in the benchmark it concentrates, whether the allocation-induced bias of the APRA phase harms long-run improvement, and what the method actually costs once generated tokens and wall-clock are accounted for alongside the execution budget. Unless stated otherwise, all numbers are LiveCodeBench (Jain et al., 2024) pass@1 at execution budget $N{=}16$ on Qwen2.5-Coder-Instruct-7B, and gains are reported as PROSE minus GRPO+CoT unless the comparator is named explicitly.

O.1 Additional Experimental Results

We report additional results for the Qwen2.5-Coder-Instruct-3B setting in Table 14. This table complements the main-body results and demonstrates that the gains of PROSE are consistent across model scales under the same execution budget. The component ordering observed at 7B (Base $<$ GRPO $<$ GRPO+CoT $<$ PVE $<$ APRA $<$ PROSE) is reproduced at 3B on all three benchmarks, and the LiveCodeBench margin over GRPO is $+3.4$ at 3B versus $+5.3$ at 7B, smaller in absolute terms and consistent with the lower absolute headroom of the smaller model (base 12.9 versus 16.9).

Table 14: Performance comparison using base model Qwen2.5-Coder-Instruct-3B. Mean $\pm$ std over 3 seeds; all methods share the execution budget $N{=}16$ and the same 1,000 optimization steps from a common base checkpoint.

	HumanEval		MBPP		LiveCodeBench	
Approach	**Pass@1**	**Pass@5**	**Pass@1**	**Pass@5**	**Pass@1**	**Pass@5**
Base	84.1±0.3	91.0±0.2	73.6±0.4	84.8±0.3	12.9±0.4	24.5±0.4
GRPO	84.9±0.4	91.5±0.3	74.8±0.4	85.4±0.3	14.2±0.5	25.8±0.5
GRPO + CoT	85.4±0.4	91.8±0.3	75.3±0.4	85.8±0.3	15.0±0.5	26.5±0.4
PVE	86.3±0.3	92.4±0.2	76.1±0.3	86.5±0.3	16.4±0.4	28.4±0.4
APRA	86.7±0.4	92.6±0.3	76.5±0.3	86.8±0.3	17.2±0.4	29.2±0.4
PROSE	**88.2±0.3**	**93.6±0.2**	**77.9±0.3**	**87.8±0.2**	**17.6±0.3**	**29.6±0.3**

O.2 Allocation Rule Ablation

To isolate the contribution of uncertainty-aware allocation itself, we compare three plan-level allocation rules under the same training setup: (i) legacy ϵ-greedy APRA (used in earlier prototypes), (ii) Thompson-sampling APRA, where scores are sampled from each plan posterior, and (iii) independent posterior-UCB APRA (PROSE). Results in Table 15 show that both posterior-based methods outperform ϵ-greedy, with UCB allocation achieving the best mean and lowest variance. The spread across the three rules (20.9 to 22.4, i.e. 1.5 pass@1 points) is smaller than the gap between flat and plan-conditioned grouping, which is the ordering the analysis predicts: the allocation rule determines how a fixed plan partition is exploited, whereas the partition itself determines how much plan-level variance is available to exploit in the first place. All three rules share the same $k{=}4$ plan partition, the same $M^{(0)}{=}2$ probe stage and the same ω_i-weighted objective; they differ only in the rule mapping probe statistics to Stage-2 counts.

Table 15: Allocation-rule ablation on LiveCodeBench (**pass@1**). All rows use k=4 semantic plans, $M^{(0)}$=2 probe rollouts per plan and N=16; only the Stage-2 allocation rule differs.

APRA allocation rule	LiveCodeBench
ϵ-greedy APRA (legacy)	20.9±0.4
Posterior Thompson APRA	22.1±0.4
PROSE (ours)	**22.4±0.3**

O.3 NOISE ROBUSTNESS UNDER EXECUTION PERTURBATION

To test robustness to noisy verifiers (e.g. flaky tests or nondeterministic runtime failures), we inject synthetic execution noise by flipping each rollout reward with probability $p_{\text{noise}} \in \{0, 0.05, 0.10\}$ during RL post-training and evaluate on the standard clean benchmark protocol. This is a direct stress test of the APRA posterior, which is estimated from exactly these corrupted rewards.

Table 16 shows that PROSE retains a margin of at least +3.3 pass@1 over GRPO+CoT, and a positive margin over legacy ϵ-greedy APRA, at every noise level: the PROSE minus GRPO+CoT margin is +4.0, +3.3 and +3.6 pass@1 at $p_{\text{noise}} = 0.00$, 0.05 and 0.10 respectively, so the ranking and the practical size of the advantage are preserved under a 10% reward-flip rate. We make no claim of *differential* robustness, and the data do not support one: from $p_{\text{noise}} = 0$ to 0.10 the absolute drop is 2.7 points for PROSE (22.4 → 19.7) and 2.3 points for GRPO+CoT (18.4 → 16.1), and the corresponding relative degradations are close to each other (approximately 12% for both). The correct reading is that plan-level posterior estimation does not amplify verifier noise relative to the flat baseline, not that it suppresses it.

Table 16: Noise-robustness evaluation on LiveCodeBench pass@1 under reward-flip probability p_{noise} applied during training. PROSE retains a margin of at least +3.3 pass@1 over GRPO+CoT at every noise level; relative degradation is comparable across methods, so no differential-robustness claim is made.

Noise level	GRPO+CoT	ϵ-greedy APRA	PROSE (ours)
$p_{\text{noise}} = 0.00$	18.4±0.4	20.9±0.4	**22.4±0.3**
$p_{\text{noise}} = 0.05$	17.8±0.4	19.6±0.4	**21.1±0.3**
$p_{\text{noise}} = 0.10$	16.1±0.5	18.2±0.4	**19.7±0.4**

O.4 HYPERPARAMETER ABLATION

We study sensitivity to the key uncertainty-allocation hyperparameters. Table 17 reports LiveCodeBench pass@1 under different uncertainty coefficients λ, exploration floors η, and plan sampling configurations (k, M). Overall, PROSE is robust to reasonable variations and does not rely on brittle tuning: every configuration in the table remains at or above 21.0, i.e. at least +2.6 over GRPO+CoT at the same budget, and the worst configuration loses 1.4 points relative to the default.

Three readings are worth isolating. First, the uncertainty coefficient has an interior optimum (λ=1.0): λ=0.5 under-explores and λ=1.5 over-weights posterior width relative to the posterior mean, costing 1.0 and 0.5 points respectively. Second, the exploration floor has an interior optimum at η=0.05, with a gentler penalty for removing the floor than for over-exploring (21.6 at η=0, 22.4 at η=0.05, 21.0 at η=0.10); removing the floor entirely does not collapse the method, which matters because η is the mechanism that protects plan diversity during APRA (Appendix O.6). Third, $M^{(0)}$=1 costs 1.3 points relative to $M^{(0)}$=2, consistent with the probe stage needing at least two executions per plan before the Beta–Bernoulli posterior is informative enough to allocate on; and k=6 at fixed N=16 costs 0.6 points, since a larger k spends more of the budget on probing and leaves fewer rollouts per plan for the code-level baseline.

Table 17: Hyperparameter ablations of uncertainty-aware PROSE on LiveCodeBench (**pass@1**). One hyperparameter is varied at a time from the default configuration (λ=1.0, η=0.05, k=4, $M^{(0)}$=2, N=16), which is the row repeated as 22.4 ± 0.3 in each block.

Hyperparameter	**Setting**	**LiveCodeBench**
Uncertainty coefficient	$\lambda = 0.5$	21.4 ± 0.4
	$\lambda = 1.0$	22.4 ± 0.3
	$\lambda = 1.5$	21.9 ± 0.4
Exploration floor	$\eta = 0.00$	21.6 ± 0.4
	$\eta = 0.05$	22.4 ± 0.3
	$\eta = 0.10$	21.0 ± 0.3
Plan sampling	$k = 4,\ M^{(0)} = 2$	22.4 ± 0.3
	$k = 4,\ M^{(0)} = 1$	21.1 ± 0.4
	$k = 6,\ M^{(0)} = 2$	21.8 ± 0.4

O.5 PLAN SEMANTIC DIVERSITY ANALYSIS

A necessary condition for plan-aware rollout reorganization to be effective is that the k sampled plans per problem are semantically diverse; if plans collapse to near-duplicates, plan-level value estimation degrades to sample-level estimation and the plan-conditioned baseline reduces to the flat one. We measure the semantic diversity of sampled plan sets by computing pairwise cosine similarity between plan embeddings (using the final-layer hidden state of the policy model at the `</plan>` token), and we call a pair a *near-duplicate* when its cosine similarity exceeds 0.85.

Table 18: Plan semantic diversity statistics over the APPS+ training set ($k = 4$ plans per problem, measured before APRA training, i.e. at the step-500 PVE→APRA switch). The $\pm$ on the first row is the dispersion *across problems* of the within-problem mean, not the standard deviation of the individual pairwise similarities, whose distribution is right-skewed and supplies the 10.0% tail above the 0.85 threshold.

Metric	**Value**
Mean pairwise cosine similarity (within-problem)	0.31 ± 0.07
% of plan pairs with cosine similarity > 0.85 (near-duplicates)	10.0%
Expected near-duplicate pairs per problem ($\binom{4}{2}$=6 pairs)	0.6
Distinct high-level strategies per problem (out of k=4)	3.4

Plans are overwhelmingly semantically distinct. The four rows of Table 18 are mutually consistent by construction, and we spell out the arithmetic so that the reader can check it: with k=4 plans there are $\binom{4}{2}$=6 within-problem pairs, so a 10.0% near-duplicate rate corresponds to an expected 0.6 near-duplicate pairs per problem; if near-duplicate pairs are predominantly disjoint—each one merging two plans into a single high-level strategy—the implied number of distinct strategies is $4 - 0.6 = 3.4$, which is what the strategy-level count measures independently. The two statistics are therefore two views of the same quantity rather than independent claims, and a materially lower near-duplicate rate would have been incompatible with a strategy count as low as 3.4. This validates the plan-diversity precondition underlying PROSE's design and confirms that plan-conditioned rollouts sample meaningfully different regions of the solution space. Appendix O.6 turns this audit from a precondition check into a test of the mechanism, by asking whether the measured gain actually tracks the measured diversity.

O.6 GAINS CONDITIONED ON PLAN DIVERSITY

The account we give of PROSE is that the benefit comes from organising rollouts around genuinely distinct latent strategies, so that the plan-conditioned baseline has plan-level variance to remove and the allocation rule has real value differences to exploit. That account makes a falsifiable prediction at the level of individual problems: the gain should be larger on problems whose k sampled plans

are more diverse, and should shrink towards zero on problems whose plans collapse. A competing account—in which the gain comes from the extra plan tokens, from a regularisation side effect of the two-stage schedule, or from any other problem-independent source—predicts no such relationship.

Diversity-conditioned gains. We bin the LiveCodeBench problems into terciles by within-problem plan diversity, measured as the mean pairwise plan-embedding cosine similarity of Appendix O.5 (a *lower* cosine means *higher* diversity), and compute the PROSE minus GRPO+CoT pass@1 difference within each tercile at the common budget $N{=}16$. Table 19 reports the result: $+2.1$, $+3.8$ and $+5.2$ pass@1 for the low-, medium- and high-diversity terciles. The ordering is monotone across the three bins and spans a factor of roughly 2.5 between the extremes, with the aggregate gain of $+4.0$ (22.4 versus 18.4) falling inside the range as it must. The same test on the near-duplicate partition gives a consistent answer: problems with at least one near-duplicate plan pair gain $+2.4$, versus $+4.1$ for problems with none.

Table 19: PROSE minus GRPO+CoT on LiveCodeBench pass@1 ($N{=}16$ for both methods), conditioned on within-problem plan diversity. Terciles are formed on the mean pairwise plan cosine similarity of Table 18; lower cosine means higher diversity. Per-subset seed standard errors are not reported, so these are point estimates on subsets of the benchmark and should be read for their ordering rather than as independently powered comparisons.

Problem subset	**PROSE − GRPO+CoT**
Plan-diversity tercile: low	$+2.1$
Plan-diversity tercile: medium	$+3.8$
Plan-diversity tercile: high	$+5.2$
Problems with ≥ 1 near-duplicate plan pair	$+2.4$
Problems with no near-duplicate plan pair	$+4.1$
All problems (reference)	$+4.0$

Both partitions point the same way, and both are consistent with the plan-structure account: the benefit tracks how much genuinely distinct strategy structure exists in the sampled plan set. This is the analysis we would expect to falsify the account if the account were wrong, and it does not. Two caveats keep the conclusion appropriately narrow. First, the subsets are not seed-error-barred, so the individual differences are point estimates; it is the monotone ordering across three bins plus the agreeing near-duplicate split that carries the evidence, not any single cell. Second, plan diversity is not de-confounded from problem difficulty: the difficulty-conditioned breakdown in Appendix O.7 orders the same way, and we do not report the joint factorisation that would separate the two, so part of the tercile effect may be difficulty operating through diversity or vice versa. The residual failure mode the analysis does not probe (all k plans collapsing onto a single strategy) is bounded above by the measured collapse rate in Table 18 rather than ruled out.

Diversity trajectory during training. A related concern is that APRA, which by design concentrates budget on high-scoring plans, might erode plan diversity over the course of training and thereby undermine its own precondition. We therefore log the diversity statistics at three points of the 1,000-step run (Table 20). Mean pairwise cosine rises from 0.29 to 0.34 and the near-duplicate rate from 9% to 13% between step 0 and step $1{,}000$. Diversity does decrease, and we report the direction plainly; the magnitude, however, is modest: a 0.05 increase in mean cosine over a full training run, ending far below the 0.85 near-duplicate threshold, and a near-duplicate rate that remains under one pair in seven. The step-500 values (0.31, 10%) coincide with the pre-APRA audit of Table 18, as they should, since step 500 is the PVE→APRA switch point.

The mechanism that bounds this erosion is the exploration floor: allocation probabilities are mixed as $q_i = (1-\eta)\tilde{q}_i + \eta/k$, so every plan retains at least η/k of the Stage-2 budget however unfavourable its posterior, and no plan can be starved out of the batch. The hyperparameter ablation (Table 17) shows the corresponding trade-off is gradual on both sides rather than catastrophic: removing the floor entirely ($\eta{=}0$) yields 21.6 and over-exploring ($\eta{=}0.10$) yields 21.0, against 22.4 at the default $\eta{=}0.05$. Even the unprotected $\eta{=}0$ configuration remains $+3.2$ over GRPO+CoT, so the cost of a partially collapsed plan set is a reduced margin, not a reversal. Taken together with the tercile result,

Table 20: Plan-diversity trajectory over the 1,000-step PROSE run. Step 500 is the PVE→APRA switch; the APRA phase is therefore the interval from step 500 to step 1,000. Lower cosine and lower near-duplicate rate mean higher diversity.

Training step	Mean pairwise cosine	Near-duplicate rate
0 (start of PVE)	0.29	9%
500 (PVE→APRA switch)	0.31	10%
1,000 (end of APRA)	0.34	13%

the picture is that PROSE's dependence on plan diversity is real and monotone in the direction the mechanism predicts, and that over the training horizons we run it is self-limiting.

O.7 WHERE THE GAINS CONCENTRATE

The plan-structure account makes a second, independent prediction about which problems should benefit. A problem that admits several genuinely distinct algorithmic strategies—choose a dynamic-programming state, choose a graph formulation, choose a greedy exchange argument—presents exactly the latent-plan structure that hierarchical advantages exploit and that posterior-UCB allocation can arbitrate between. A problem whose difficulty lies almost entirely in careful implementation of a single evident approach presents very little such structure, and the plan-conditioned baseline has correspondingly little plan-level variance to remove. The prediction is therefore an ordering over problem types, stated before the breakdown was computed, not a fitted description of it.

Table 21 reports the PROSE minus GRPO+CoT pass@1 difference, at the common budget $N{=}16$, split by LiveCodeBench difficulty tier and by tagged problem type. By difficulty, the gain is $+2.3$ on easy, $+4.0$ on medium and $+5.1$ on hard problems. By type, the ordering is dynamic programming $+5.0$, graph $+4.8$, greedy $+4.3$, string $+3.0$ and implementation $+2.2$. Both breakdowns bracket the aggregate $+4.0$, and the type ordering is the one the account predicts: the three strategy-rich categories occupy the top three positions and implementation-dominated problems the bottom. String problems sit between the two groups; we do not have a mechanism-level account of their intermediate position and do not offer one.

Table 21: PROSE minus GRPO+CoT on LiveCodeBench pass@1 ($N{=}16$ for both methods), broken down by difficulty tier and by problem type. Problem-type tags were assigned by an LLM and spot-checked by the authors (see the AI-use statement); tags are single-label, so a problem admitting both a DP and a greedy formulation is counted once, under the tag the model assigned. As in Table 19, per-subset seed standard errors are not reported and the evidence is carried by the ordering rather than by individual cells.

Conditioning	Subset	PROSE − GRPO+CoT
Difficulty tier	Easy	$+2.3$
	Medium	$+4.0$
	Hard	$+5.1$
Problem type	Dynamic programming	$+5.0$
	Graph	$+4.8$
	Greedy	$+4.3$
	String	$+3.0$
	Implementation	$+2.2$
All problems (reference)		$+4.0$

Two limitations bound what this breakdown establishes. The tagging is automatic: an LLM assigned a single algorithmic-category label to each problem and the authors spot-checked a sample of the assignments rather than auditing all of them, so the type partition carries label noise of an unquantified magnitude. And the difficulty and type conditionings are correlated in this benchmark (hard problems are disproportionately DP and graph problems), so the two panels of Table 21 are not independent tests, and neither is independent of the diversity conditioning in Appendix O.6. What the three

analyses jointly support is that the gain is concentrated where the mechanism says it should be concentrated, on problems with real strategic branching; what they do not support is a decomposition of the effect into orthogonal difficulty, type and diversity components.

O.8 Long-Horizon Behaviour of APRA

The theory covers the PVE stage exactly: under uniform allocation the plan-conditioned advantage is unbiased, and Theorem 4.1 gives its variance advantage over the flat comparator. The APRA stage deliberately leaves that regime. Non-uniform allocation makes the allocation-weighted plan-value estimator biased, and Theorem F.1 bounds that bias, quadratically in the plan-value spread, with explicit dependence on the posterior error $\varepsilon_{\max}$ and the exploration floor η. A bound is not a guarantee that the bias is harmless in practice over a long run, so we report three independent signals, all pointing the same way.

(i) Training reward curves. Figure 4 shows that a PVE-only run plateaus around step 700 while the full PROSE schedule continues to improve after the step-500 switch. This signal is the weakest of the three and we treat it as such: the plotted quantity is the batch mean reward, which posterior-UCB allocation mechanically inflates by concentrating rollouts on plans it currently believes are good, independently of any improvement in the policy. The curve is suggestive of a slope change at the switch; it is not, on its own, evidence of a better policy.

(ii) Held-out checkpoint evaluations across the APRA phase. This signal is free of that confound, because it evaluates saved checkpoints on the clean benchmark protocol rather than reading the training batch. Table 22 reports LiveCodeBench pass@1 at steps 600, 800 and 1,000, all inside the APRA phase: 21.3, 21.9 and 22.4. The sequence is monotone increasing across the entire second half of training, and its endpoint is the headline 22.4. Whatever bias the allocation rule introduces into the plan-value estimates, it does not accumulate into a degradation of held-out performance over 500 steps of APRA; the direction of improvement is preserved throughout.

Table 22: LiveCodeBench pass@1 at checkpoints inside the APRA phase of the 1,000-step PROSE schedule (APRA is active from step 500 onwards). Evaluation uses the clean benchmark protocol, so these numbers are not subject to the batch-reward confound affecting Figure 4.

Checkpoint (APRA phase)	LiveCodeBench pass@1
Step 600	21.3
Step 800	21.9
Step 1,000 (final)	22.4

(iii) Posterior corruption stress test. The final signal attacks the posterior directly. Reward-flip noise during training corrupts exactly the Beta–Bernoulli statistics that APRA allocates on, so if allocation-induced bias were the fragile component of the method, this is the perturbation that should expose it. It does not: as reported in Table 16, at $p_{\text{noise}}=0.10$ PROSE reaches 19.7 against GRPO+CoT's 16.1, a margin of $+3.6$ against $+4.0$ in the noiseless setting. The lead is retained essentially in full under a corruption rate an order of magnitude above what a well-maintained test suite would produce.

None of these three is individually decisive: (i) is confounded, (ii) reports point estimates without seed dispersion, and (iii) tests robustness rather than long-horizon accumulation. Their value is that they fail in different ways. Taken together they support the modest claim we make: over the training horizons we run, the allocation-induced bias bounded by Theorem F.1 does not interfere with long-run policy improvement. We do not claim this extends to horizons substantially beyond 1,000 steps, where the posterior has had more opportunity to entrench an early misallocation; that regime is untested here.

O.9 COMPUTE ACCOUNTING

Execution budget, generated tokens and wall-clock are three distinct costs, and PROSE does not stand in the same relation to the baselines on all three. This appendix reports each separately and never as a composite, expanding the main-text compute-accounting summary (Table 4).

Aggregate token accounting. From the aggregate training logs of the 1,000-step runs, generated tokens per problem are 6.1k for GRPO, 6.5k for GRPO+CoT and 6.3k for PROSE (Table 23). PROSE at $N{=}16$ therefore generates *fewer* tokens per problem than GRPO+CoT at the same execution budget. Relative to vanilla GRPO, which generates no plan text at all, the logged overheads are $\approx 3\%$ for PROSE and $\approx 7\%$ for GRPO+CoT; both figures are computed from counts rounded to two significant figures and should be read as approximate. This is consistent with the 5–8% token-overhead class reported for plan-generating methods, with PROSE at or below its lower end. The aggregate is the basis for every claim we make about token cost; the case studies below are illustrative of the mechanism behind it.

Table 23: Compute accounting at the common execution budget $N{=}16$, with the three cost axes kept separate. Tokens are generated tokens per problem from aggregate training logs. Wall-clock is for the full 1,000-step run on identical hardware (8×H800); the two PROSE comparisons are the measured ones, and the GRPO+CoT entry is the value they imply (1.050/1.018).

Method	Exec. budget N	Plan derivations	Tokens / problem	Wall-clock
GRPO	16	0	6.1k	reference
GRPO + CoT	16	16	6.5k	+3.1% (implied)
PROSE	16	4	**6.3**k	+5.0% vs. GRPO

Matched-success case studies. Aggregate token counts mix together problems on which the methods succeed at different rates, and a method that solves more problems may generate more tokens for that reason alone. To remove that coupling we isolate problems on which both methods achieve the *same* number of successful rollouts out of 16. Table 24 reports two such cases. At 12/16 successes for both methods, GRPO+CoT generates 6,436 tokens (612 planning, 5,824 code) against PROSE's 6,108 (153 planning, 5,955 code), 5.1% fewer in total. At 10/16 successes for both, the totals are 6,281 against 6,037, or 3.9% fewer. These are two individual problems, presented to make the mechanism legible rather than to establish the magnitude; the aggregate row of Table 23 is what the conclusion rests on.

Table 24: Matched-success token case studies: individual problems on which GRPO+CoT and PROSE achieve the same number of successful rollouts out of $N{=}16$, so that token counts are compared at equal outcome. The planning/code split was logged for Case 1 only. These two cases are illustrative; the aggregate accounting in Table 23 is the basis of the token-cost conclusion.

Case	Method	Planning tok.	Code tok.	Total
Case 1 (12/16 successes)	GRPO + CoT	612	5,824	6,436
	PROSE	153	5,955	**6,108** (−5.1%)
Case 2 (10/16 successes)	GRPO + CoT	—	—	6,281
	PROSE	—	—	**6,037** (−3.9%)

Why the plan-conditioned method is the cheaper one. The result is initially counter-intuitive—PROSE generates plan text and still spends fewer tokens than the baseline that also generates plan text—but the mechanism is simple amortisation. PROSE samples $k{=}4$ plans *once* per problem and then *shares* each plan across the multiple code rollouts allocated to it, so planning cost scales with k, not with N. GRPO+CoT re-derives an inline plan inside every one of its 16 rollouts, so its planning cost scales with N. Case 1 exhibits this exactly: $612/16 = 38.25$ planning tokens per derivation for GRPO+CoT and $153/4 = 38.25$ for PROSE. The per-derivation cost is identical—both methods write a plan of the same length, consistent with the 35–40 plan-token figure reported for the method—and

the entire difference is that one method performs 16 derivations and the other performs 4. PROSE's code-token count is slightly higher (5,955 versus 5,824), which is the expected direction: rollouts conditioned on an explicit shared plan tend to produce marginally longer programs, and the net effect is still a token saving.

Wall-clock. Over the full 1,000-step run on identical hardware, PROSE costs $+5.0\%$ wall-clock over vanilla GRPO and $+1.8\%$ over GRPO+CoT. Essentially all of this is generation: the Beta–Bernoulli posterior updates and the softmax-UCB allocation operate on $k{=}4$ scalar plan statistics per problem and are closed-form, so their cost is negligible against forward passes over thousands of generated tokens; plan clustering is an embedding lookup plus greedy cosine assignment over k items. The $+5.0\%$ wall-clock figure exceeds the $\approx 3\%$ token overhead relative to GRPO because plan generation is a short, separately batched decode phase that amortises less well than the long code-generation phase, not because of any new super-linear component. For scale, the reference $8\times$H800 configuration completes a 1,000-step run in approximately 60 hours.

Conservative control: granting the baseline the overhead as extra rollouts. The strictest form of the fairness question is to hand the baseline the entire token overhead in the currency it actually cares about: additional executions. Granting the worst case of the 5–8% overhead class to GRPO+CoT corresponds to an execution budget of $16 \times 1.08 \approx 17.3$, that is, at most two executions beyond the matched $N{=}16$. The budget-scaling ladder settles that case with room to spare: GRPO+CoT requires $N{=}32$ to reach 21.1 (Table 25), so even at double the execution budget it does not overtake PROSE at $N{=}16$ (22.4 ± 0.3), and two additional executions cannot close a gap that sixteen additional executions leave open. The token accounting above and this control address the same concern from opposite directions, and they agree.

P EXTENDED EVALUATION NOTES

This appendix holds evaluation material moved out of Section 5 under the main-text page limit, in the order in which it appears below: secondary readings of the main results table, the budget-scaling ladder and curve, both design-effect routes of the secondary sanity check, the progressive plan/allocation ablation table, the cross-domain (GSM8K) and cross-family (DeepSeekCoder-7B) result tables, and the training-reward diagnostic. No claim here is new: every number the main text relies on is quoted in Section 5, and the tables below present the same measurements at length.

P.1 SECONDARY READING OF THE MAIN RESULTS TABLE

Three observations on Table 2 that the main text does not have room for.

First, HumanEval and MBPP start near ceiling, and the ordering there is not the LiveCodeBench ordering in miniature: over the Base checkpoint, GRPO adds $+0.3$ and $+0.2$ pass@1 while PROSE adds $+3.1$ and $+3.3$. The reorganization therefore does something on benchmarks where the reward signal is dense and the headroom small, and the effect is not confined to the sparse-reward regime that LiveCodeBench represents. The absolute margins are nonetheless compressed by the ceiling, and we do not read the near-saturated benchmarks as independent confirmation of the size of the LiveCodeBench effect.

Second, the LiveCodeBench pass@5 margin of PROSE over GRPO (37.1 versus 29.5, i.e. $+7.6$) exceeds its pass@1 margin ($+5.3$). A method that merely sharpened the policy onto a single strategy would compress pass@5 relative to pass@1; the opposite ordering is what one expects when the reorganization preserves several distinct plans per problem and improves the credit assigned to each. This is consistent with, though not a test of, the plan-diversity analysis of Appendix O.6.

Third, the external checkpoints in the bottom block (Dream-Coder (Xie et al., 2025), AceCoder (Zeng et al., 2025) and OlympicCoder (Open R1 / Hugging Face, 2025)) are evaluated under our protocol as absolute reference points only. They are neither budget- nor training-matched: their base models, training data, token budgets and optimization horizons differ from ours and from each other. No claim in this paper rests on them, and they are reported only so that the absolute scale of our numbers can be located against published artefacts.

P.2 BUDGET-SCALING LADDER AND CURVE

Table 25: Budget scaling on LCB pass@1 (mean ± std over 3 seeds); columns give the execution budget N. Summarised in Section 5, where every comparison drawn from this table is quoted in the text.

Method	N=8	N=16	N=32
GRPO	15.8±0.4	17.1±0.4	19.8±0.5
GRPO + CoT	17.2±0.4	18.4±0.4	21.1±0.4
PROSE	**20.1±0.3**	**22.4±0.3**	**25.8±0.3**

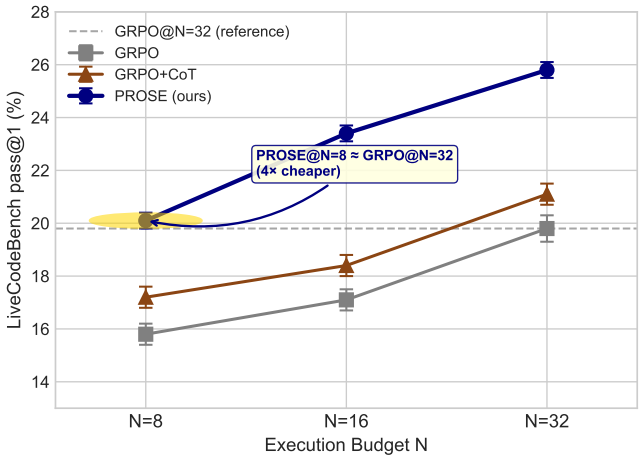

Figure 3: Budget scaling on LCB pass@1 against the execution budget N: the same three measured points per method as Table 25, plotted. TOST equivalence under the pre-specified δ=1.0 margin is claimed only against *vanilla* GRPO; against GRPO+CoT the surviving claim is strict superiority at half the budget (Section 5).

Figure 3 plots Table 25 and adds nothing to it; it is kept because the horizontal reading is easier to see than to state. The leftward displacement of the PROSE curve quoted in Section 5 is measured against the GRPO+CoT curve under a log-linear fit through the three measured budgets $N \in \{8, 16, 32\}$, and only the displacement at the smallest budget is read inside the measured range. Over the accuracy band $[20.1, 21.1]$, where both curves are measured, it is 2.96–3.09×, i.e. about 3×. GRPO+CoT tops out at 21.1 at N=32, so matching PROSE at N=16 (22.4) or at N=32 (25.8) instead requires projecting that fit to N≈45 and N≈107; the 2.8× and 3.3× endpoints of the quoted range are those two extrapolations, and we report the range as an indicative ordering rather than as a measured budget multiplier. Two cautions carry over from Section 5. The equivalence statement in the caption is scoped to execution budget and to vanilla GRPO; PROSE@N=8 (20.1) lies 1.0 pass@1 *below* GRPO+CoT@N=32 (21.1), so no 4× reading survives against the stronger baseline. And the horizontal axis counts executions only: token and wall-clock costs are accounted separately in Table 4.

P.3 THE TWO DESIGN-EFFECT ROUTES OF THE SECONDARY SANITY CHECK

Section 5.3 translates the gradient design effect through the measured GRPO+CoT budget curve as a *secondary sanity check*. The theorem-level quantitative prediction of Theorem 4.1(a) is the advantage-variance ratio, and it is tested directly: R_{same}=0.262 predicted against 0.26 measured on the stored k=4, M=4 batch. The budget-curve translation below is a consistency check on the ordering of two routes rather than a theorem-level prediction of pass@1, and no claim in the paper depends on its numerical value.

Gradient-level route. At $\widehat{\rho}_h^{\text{flat}}$=0.56 and M=4, the gradient design effect is $D_{\text{eff}} = 1 + (M-1)\widehat{\rho}_h^{\text{flat}}\rho_r \approx 2.21$, i.e. $\log_2(2.21) \approx 1.14$ effective budget doublings. The GRPO+CoT budget curve in Table 25 has slope $(21.1-17.2)/2 = +1.95$ pass@1 per doubling, so the translated estimate is $\approx +2.2$ pass@1, against the $+1.9$ actually observed for the hierarchical-advantage step (CoT $18.4 \rightarrow$ PVE 20.3). The exact finite-batch design effect at this operating point is 1.68 (Remark H.4), which translates to $\approx$+1.5 rather than +2.2; the ordering of the two routes, which is what the check tests, is unchanged.

Reward-level route. Substituting the reward-level design effect D_r=3.16 for D_{eff} gives $\log_2(3.16) \approx 1.66$ doublings and a translated estimate of +3.2 pass@1. This over-predicts the observed +1.9, and it must: D_r is exactly the ρ_h^{flat}=1 upper bound of D_{eff}, so by Theorem 4.1(c) the reward-level route is an upper bound on the gradient-level one rather than an estimate of it. The *ordering* of the two routes—gradient-level route close to the observation, reward-level route strictly above it—is the check being performed. Neither route turns the theorem into a prediction about pass@1, and neither is offered as one.

P.4 Progressive Plan and Allocation Ablation

Table 26: Progressive ablation on LCB pass@1 at N=16, adding one mechanism at a time. Not a complete 2 × 2: "None (flat)" is plain GRPO, not a Random | Uniform control, so the +1.2 marginal mixes allocation rule with grouping structure.

Plan	Alloc.	LCB
None (flat)	Uniform	17.1±0.4
Random	UCB	18.3±0.4 (+1.2)
Semantic	Uniform	20.6±0.4 (+3.5)
Semantic	**UCB**	**22.4±0.3** (+5.3)

Table 26 is the table summarised in Section 5. Semantic plans under uniform allocation capture +3.5 of the +5.3 (66%); posterior-UCB over random groups adds +1.2; UCB applied to semantic plans adds a further +1.8 on top of the uniform-allocation row. The reading is that allocation helps most exactly when the groups it allocates over correspond to latent plans with informative pass-rate variation: the same condition that the variance argument of Theorem 4.1 needs. Random groups carry near-zero ICC (≈ 0.05), and the +1.2 they buy is therefore a bound on what generic adaptive allocation contributes, about 23% of the headline +5.3.

Relation to the PVE row of Table 2. The Semantic | Uniform cell above (20.6±0.4) and the PVE row of Table 2 (20.3±0.3) are reported from separate measurement tables; their reported intervals, $[20.2, 21.0]$ and $[20.0, 20.6]$, overlap on $[20.2, 20.6]$, so the 0.3 difference sits inside the seed dispersion each table reports. We record the arithmetic here so that the two figures are not read against each other as a discrepancy; no claim in the paper rests on the comparison.

Design caveat. The design is not a complete 2 × 2. The "None (flat)" cell is plain GRPO (no grouping at all) rather than a Random | Uniform control, so the +1.2 marginal attributed to the allocation rule is confounded with the presence of grouping structure. A true Random | Uniform cell would separate the two, and we do not have one. The main text states this caveat wherever the +1.2 is used.

Interaction term. The combined +5.3 exceeds the sum of the two marginals (+1.2 and +3.5) by +0.6. With 3 seeds, the contrast has standard error $\sqrt{(0.09 + 3 \times 0.16)/3} = 0.44$, giving $t \approx 1.38$: the interaction is not resolvable at this seed count. We therefore claim only that the mechanisms *compose at least additively*, and that we find no evidence of sub-additivity. We do not claim a synergistic interaction, and the +0.6 should not be read as one.

Warm-up within the allocation step. Inside the +2.1 allocation step, the +1.4 gap between APRA and full PROSE (21.0 → 22.4 in Table 2) isolates the uniform Stage-1 warm-up: cold APRA allocates

from noisy probe statistics, and concentrating executions on a plan whose posterior is still dominated by its prior wastes budget on the wrong plan often enough to cost most of the step.

P.5 CROSS-DOMAIN TRANSFER: GSM8K

Table 27: Cross-domain results on GSM8K (Cobbe et al., 2021): pass@1 / pass@5 with Qwen2.5-Instruct-7B, mean ± std over 3 seeds at N=16.

Method	Pass@1	Pass@5
GRPO	92.0±0.4	97.0±0.2
GRPO + CoT	92.4±0.4	97.2±0.2
PROSE	**93.6±0.3**	**97.6±0.2**

Latent-plan reward correlation is not specific to code: it arises whenever rollouts for the same input share a latent strategy. On GSM8K (Cobbe et al., 2021) with Qwen2.5-Instruct-7B (Qwen Team, 2024), under hyperparameters identical to the code setting, a "plan" is a solution strategy, and the reward-level ICC measured on 500 held-out problems is $\rho_{\text{math}} \approx 0.48 \pm 0.16$ ($D \approx 2.44$), below the code value (ρ_r=0.72, D_r=3.16). Table 27 gives the corresponding gains: +1.6 pass@1 over GRPO and +1.2 over GRPO+CoT.

This is smaller than the +5.3 on LiveCodeBench, in the direction that Eq. (2) predicts from the smaller design effect, and it is further compressed by the ceiling: all three methods sit above 92%, where a fixed relative error reduction buys few absolute points. Measured like for like against GRPO, the +1.6 here removes 20% of the residual error (1.6/8.0), against 6.4% (5.3/82.9) for the +5.3 on code. On that scale the mathematics reduction is the larger of the two, which is what a ceiling-compressed gain looks like: the absolute number is small because the error budget is small, not because the reorganization does less work per unit of error available to it. The +31.0% figure quoted for code in Section 5 is a different quantity, a relative pass@1 gain of 5.3/17.1, and is not comparable with an error reduction; we state both scales here because the absolute and relative orderings genuinely disagree. Two limits on what this result supports. First, it is an ordering test only: the gradient-level correlation ρ_h^{flat} was measured on code and not on GSM8K, so we can say that a smaller reward-level design effect accompanies a smaller gain, but not that the gradient-level mechanism has been verified in this domain. Second, with a single mathematics benchmark at ceiling, the comparison cannot separate the design-effect explanation from the ceiling explanation. Both predict the smaller absolute gain, and the relative-error comparison above, taken alone, is accounted for by the ceiling; we therefore do not apportion the difference between the two explanations, and we do not claim the smaller absolute gain as confirmation of the design-effect route.

P.6 CROSS-FAMILY TRANSFER: DEEPSEEKCODER-7B-INSTRUCT

Table 28: Generalization to DeepSeekCoder-7B-Instruct (Guo et al., 2024) at N=16 (std ≤ 0.5 throughout, omitted).

Approach	HumanEval	MBPP	LiveCodeBench
Base	83.1	80.4	14.8
GRPO	84.2	81.3	16.2
GRPO+CoT	85.1	82.0	17.6
PROSE	**87.4**	**83.5**	**21.3**

Table 28 repeats the protocol on a different model family. PROSE yields +3.7 LiveCodeBench pass@1 over GRPO+CoT (17.6 → 21.3), mirroring the +4.0 obtained on Qwen2.5-Coder-Instruct-7B, and +5.1 over vanilla GRPO. The diagnostic quantity transfers with it: the reward-level ICC on this policy is $\rho_r \approx 0.69$, close to the 0.72 measured on Qwen, so Eq. (2) predicts a comparable design effect and no retuning of k, $M^{(0)}$, λ or η was performed for this family. The practical reading is that the ICC is a property of the task-and-policy pair that can be measured once on a new family before deciding whether the reorganization is worth applying, rather than a constant inherited from our setting.

P.7 Training-Reward Dynamics

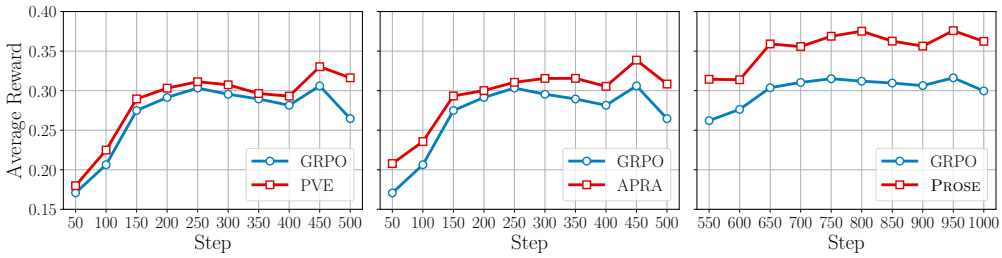

Figure 4: Training reward curves. PVE stabilizes plan-level signal in warm-up where GRPO oscillates; APRA activates at step 500 and changes the slope; a PVE-only run plateaus near step 700 while the full schedule keeps improving. **Caveat:** posterior-UCB mechanically raises PROSE's batch-mean reward by concentrating executions on high-posterior plans, so this is not an unbiased policy comparison; an unbiased estimate would use only the uniformly allocated Stage-1 probe rollouts. Read it as a training-dynamics diagnostic; all policy comparisons in this paper rest on held-out pass@1.

Figure 4 is a diagnostic, not a comparison. The batch-mean training reward plotted for PROSE is computed over rollouts that posterior-UCB has deliberately concentrated on plans with high posterior means, which raises that mean mechanically and independently of any policy improvement. The confound is structural: it is the allocation rule working as designed. An unbiased comparison at the same points in training would restrict the estimate to the uniformly allocated Stage-1 probe rollouts, which are drawn under the same uniform rule as the baselines' rollouts. What the figure does support is the qualitative shape of the schedule—the stabilisation of plan-level signal during warm-up, the slope change when APRA activates at step 500, and the plateau of a PVE-only run near step 700. Every policy-level claim in Section 5 rests on held-out pass@1 instead, and the monotone APRA checkpoint ladder of Appendix O.8 ($21.3/21.9/22.4$ at steps $600/800/1{,}000$) is the held-out counterpart of the slope change visible here.

Q Extended Discussion of Positioning

This appendix holds material moved out of Sections 1 and 2 for length: the scope of the diagnosis outside code generation, an explicit statement of which parts of PROSE's gain the theory covers, and the per-method comparisons against the closest rollout-level and process-level methods. The gradient-level design effect is developed in Appendix D.4.

Q.1 Scope of the diagnosis beyond code generation

Nothing in the diagnosis of Section 3.2 is specific to programs. It requires only that rollouts within a group share a latent variable that predicts reward, so that rewards follow a one-way random-effects mixture with $\rho_r > 0$; the structure it exploits is therefore available in any grouped-sampling RL setting whose rollouts share latent strategies, such as mathematical reasoning or tool use.

We state this as a hypothesis with one domain of supporting evidence rather than as a claim of generality. On GSM8K the measured ICC is $\rho_r \approx 0.48$ ($D = 2.44$ at $M{=}4$), lower than the $\rho_r \approx 0.72$ ($D_r = 3.16$) measured on APPS+, and the observed gain is correspondingly smaller: $+1.6$ pass@1 against $+5.3$ on code. That comparison is an ordering test only, and it is additionally compressed by a ceiling effect, since all methods sit above 92%; Appendix P.5 gives the residual-error arithmetic and states why the absolute and relative orderings disagree. The score-correlation proxy ρ_h^{flat} was *not* measured on GSM8K, so the gradient-level route is untested there.

Two settings are outside what we have evidence for. Repository-level and agentic code generation would require rethinking what a "plan" is at task granularity, and what the unit of clustering should be when a single episode spans many files or tool calls; we regard that as the natural next paper rather than as implied coverage of the present one. Longer-horizon settings in which the latent strategy changes within an episode also fall outside the single-plan-per-rollout model assumed throughout,

and we make no claim about them; the long-horizon study of Appendix O.8 answers a different question, whether allocation bias accumulates over $1{,}000$ training steps.

Q.2 What the theory covers, and what it does not

PROSE is a plan-aware allocation-and-credit method whose hierarchical-advantage core is characterized, and directly predicted, by the measured ICC. It is not a method whose entire gain the theory explains, and we do not present it as one. Against the $+5.3$ total over GRPO:

- $+1.3$ from explicit plan generation (GRPO $\rightarrow$ GRPO+CoT) sits outside the theory entirely and is taken as given; it is the component shared with inference-time planning methods.
- $+1.9$ from the hierarchical advantage is the component Theorem 4.1(a) and (b) cover, and the only one subjected to a direct quantitative test.
- $+2.1$ from posterior-UCB allocation with ω-weighting and warm-up is covered only by a bias bound; its magnitude is empirical.

The optimality statement carries a standing qualifier: the hierarchical advantage is UMVU *within the symmetric subtractive class*. Process-level and partial-credit rewards leave that class by construction, so the optimality claim does not extend to them; this is also the precise sense in which PROSE and the process-supervision line below are not competing for the same result.

Q.3 Per-method comparison

Rollout selection and reweighting (REINFORCE-ADA, XRPO). REINFORCE-ADA (Xiong et al., 2025) and XRPO (Bamba et al., 2025), like Xu et al. (2025); Zheng et al. (2025); Yao et al. (2025), act on *individual rollouts* within a group whose composition is already fixed: they reweight, prune or resample using observable per-rollout statistics—difficulty, reward density, redundancy—and never represent the latent variable that makes two rollouts covary in the first place. Two consequences follow. First, their selection signal is a symptom of the latent-plan reward correlation rather than the correlation itself, so it cannot say at what granularity a baseline should be computed. Second, because the group's composition is fixed before selection acts, they cannot spend extra executions on a plan that looks promising and stop spending them on one that does not. PROSE operates one level up: it makes the plan explicit, estimates plan values at cluster granularity, and reallocates budget across rather than within plans, tying the reorganization quantitatively to the measurable ICC.

That the distinguishing ingredient really is the plan-level clustering, and not simply extra structure in the pipeline, is testable. A post-hoc control that extracts plans from GRPO+CoT rollouts by an auxiliary LLM and applies the hierarchical advantage to the extracted clusters, without any forward plan conditioning, reaches 19.7 LCB pass@1—recovering $+1.3$ of the $+1.9$ hierarchical-advantage step, with a residual gap of 0.6 to forward PVE at 20.3 (Appendix M).

Process-level supervision and credit assignment (Posterior-GRPO, PRLCoder, ARIA). Posterior-GRPO (Fan et al., 2025) uses posterior signals for token-level credit assignment, PRLCoder (Ye et al., 2025) supplies fine-grained process supervision, and ARIA (Yang et al., 2025) aggregates rewards at the intention level. All three change what is rewarded inside a rollout; PROSE changes which rollouts exist and over what set the baseline is pooled, leaving the reward function itself untouched. The distinction matters formally: acting on credit within a rollout is exactly what takes an estimator outside the symmetric subtractive class of Theorem 4.1(b), so the UMVU result does not transfer to those methods and they are not in competition with it. The same applies to the broader reward-design line (rubric-based reward models (Gunjal et al., 2025) and the security-aware partial credit of SecureCodeRL (Sijwali & Saha, 2026)), which extends the reward beyond execution outcomes. Posterior-GRPO is the closest of these in mechanism, since it too maintains a posterior; the difference is what the posterior is over. Its posterior scores token spans within a rollout; PROSE's scores plans and sets how many rollouts each plan receives, which is why only the latter changes the group's composition and hence the variance of the baseline.

Planning and structured reasoning. PLANSEARCH (Lei et al., 2025) shows that natural-language plans improve search effectiveness; self-planning (Jiang et al., 2024) and planning-before-coding (Zhang et al., 2025) split generation into planning and implementation stages; tree-structured

search (Yao et al., 2023) and multi-agent systems (Islam et al., 2024) pair planning with collaboration. These methods improve individual generations at inference time and leave the training-time organization of rollouts unchanged. The training-time analogue of plan prompting alone is GRPO+CoT, which we include as a baseline at every execution budget; in the attribution ladder above it accounts for $+1.3$ of PROSE's $+5.3$ gain, i.e. the planning literature's contribution is present in our strongest baseline rather than absent from the comparison.

RL with execution feedback. AlphaCode (Li et al., 2022), CodeRL (Le et al., 2022) and RLTF (Liu et al., 2023) establish execution-feedback optimization as a route to competitive functional correctness, and StepCoder (Dou et al., 2024) and RLEF (Gehring et al., 2024) mitigate reward sparsity through curricula or richer feedback. Auxiliary test-case synthesis (Jiang et al., 2025; Wang et al., 2025) and knowledge adaptation (Wu et al., 2025) instead densify or update the signal itself. PROSE is orthogonal to both groups: it leaves the reward unchanged and restructures the group over which the baseline is computed, so it composes with any of them that keeps the reward binary and execution-derived.

R HYPERPARAMETERS, INFRASTRUCTURE, AND REPRODUCIBILITY DETAILS

This appendix gives the configuration used for every in-house run reported in Section 5. Table 29 lists the method-level PROSE configuration (plan sampling and budget bookkeeping, decoding, APRA allocation, and the two-stage schedule), and Table 30 the optimiser and RL objective, the reward, the evaluation and reporting protocol, and the training infrastructure; everything in Table 30 is shared verbatim by every method. Table 31 then separates what every method shares from the (small) set of settings that distinguishes each method, so that the ablation ladder Base $\rightarrow$ GRPO $\rightarrow$ GRPO+CoT $\rightarrow$ PVE $\rightarrow$ APRA $\rightarrow$ PROSE can be read as a sequence of single-mechanism edits on a fixed substrate.

Decoding parity across methods. Sampling temperature is a confounder for both pass@1 and exploration dynamics, so we fix it once and never vary it by method. All rollouts — for every method, in both the plan-generation and code-generation steps, during training and at evaluation — are sampled at temperature 0.7 with top-p=1.0, from the same decoding backend and under the same evaluator. No method, PROSE included, is run at a different generation temperature from any other, and no reported gain is attributable to a decoding difference. The third temperature appearing in Table 29, τ_{alloc}=0.7, is *not* a sampling temperature: it is the softmax temperature of the Stage-2 budget allocation over posterior-UCB scores u_i, it applies to PROSE/APRA only because no other method allocates non-uniformly, and it never touches token sampling. The numerical coincidence of the three values is not a coupling: τ_{alloc} is a free parameter of the allocation rule and can be changed without touching decoding. We did not ablate it, and it is held at 0.7 in every reported run.

Budget bookkeeping and the N=8 configuration. The execution budget decomposes as $N = k\,M^{(0)} + N_{\text{stage2}}$: Stage 1 spends $k\,M^{(0)}$ executions probing each of the k plans uniformly, and Stage 2 reallocates the remaining N_{stage2} by posterior-UCB. At the default operating point (N=16, k=4, $M^{(0)}$=2) this is 8 probe executions and 8 reallocated executions, and the balanced comparator used throughout the theory is $M = N/k = 4$. The budget-scaling study in Section 5 evaluates PROSE at N=8 as well, where the default $M^{(0)}$=2 would consume all eight executions in Stage 1 and leave APRA nothing to reallocate. The N=8 runs therefore use k=4 with $M^{(0)}$=1: four probe executions and four reallocated executions, so the reported N=8 row does exercise the posterior-UCB mechanism and is not a degenerate PVE-only run. Two consequences are worth stating plainly. First, the balanced comparator at N=8 is $M = N/k = 2$, not 4, so the variance-reduction factor predicted by Theorem 4.1(a) is correspondingly smaller at that budget. Second, $M^{(0)}$=1 gives each plan a single Bernoulli probe before reallocation, i.e. a $\text{Beta}(1,1)$ posterior updated by one observation, which is the noisiest posterior any configuration in this paper allocates from; the matched ablation at N=16 (Table 17, k=4, $M^{(0)}$=1: 21.1±0.4 versus 22.4±0.3 at $M^{(0)}$=2) quantifies that cost as -1.3 pass@1 at fixed budget. The N=8 result should be read with that penalty already absorbed into it.

What is shared and what differs across methods. Table 31 is organised so that the shared substrate is stated once and each method is defined only by its delta from it. Every in-house method

Table 29: PROSE method configuration: plan sampling and budget, decoding, APRA allocation, and the two-stage schedule. Decoding settings are identical for every method compared in this paper, in both training and evaluation; τ_{alloc} is a budget-allocation softmax temperature and not a sampling temperature. Optimiser, reward, reporting, and infrastructure settings are in Table 30.

Hyperparameter	Value	Description and scope
Plan sampling and execution budget		
k	4	Plans sampled per input
N	16	Total execution budget per input, $\sum_{i=1}^{k} M_i = N$
$M^{(0)}$	2	Stage-1 uniform probe budget per plan; $k\,M^{(0)}$=8 probe executions, 8 left for Stage-2 reallocation
M	$N/k = 4$	Balanced per-plan size; the comparator M in Theorem 4.1
$(k, M^{(0)})$ at N=8	$(4, 1)$	Budget-scaling study only: 4 probe + 4 reallocated executions, so APRA remains active
Decoding — identical for all methods, training and evaluation		
τ_{plan}	0.7	Plan-generation temperature; **standardised across all methods**
τ_{code}	0.7	Code-generation temperature; **standardised across all methods**, and the pass@1/pass@5 evaluation decoding temperature
top-p	1.0	Nucleus parameter, all methods, both stages, training and evaluation
*APRA allocation (*PROSE*/APRA only; all other methods allocate uniformly)*		
Allocation rule	Posterior-UCB	Stage-2 score $u_i = \mu_i + \lambda\sigma_i$ from the plan posterior
(α_0, β_0)	$(1, 1)$	Beta prior for the Bernoulli execution reward
λ	1.0	Uncertainty-bonus (UCB) coefficient
η	0.05	Exploration floor: $q_i = (1-\eta)\tilde{q}_i + \eta/k$
τ_{alloc}	0.7	Softmax temperature over u_i in the allocation, $\tilde{q}_i \propto \exp(u_i/\tau_{\text{alloc}})$; not a sampling temperature
ω_i	$M_i/(N/k)$	Allocation-aware gradient weight (APRA); $\omega_i \equiv 1$ during PVE
Rounding	Largest remainder	Integerisation of $N_{\text{rem}}q_i$ into Stage-2 counts $M_i^{(1)}$, where N_{rem}=$N-kM^{(0)}$; $M_i = M^{(0)} + M_i^{(1)}$
ε	10^{-6}	Numerical stabiliser for uncertainty normalisation
Two-stage schedule		
T_{PVE}	500	PVE warm-up steps (uniform allocation, $\omega_i \equiv 1$), steps 0–500
T_{APRA}	500	APRA steps (posterior-UCB allocation, ω_i-weighted), steps 500–1,000
Total steps	1,000	Identical for every method, from a common base checkpoint
LR warm-up	None	Constant learning rate throughout; the only warm-up is the algorithmic T_{PVE} stage

uses the same base checkpoint, the same APPS+ training split, the same 1,000 optimisation steps, the same optimiser and learning rate, the same batch sizes, the same KL and clipping coefficients, the same total execution budget N=16 per input, the same decoding configuration, the same seeds, and the same evaluator. The deltas are correspondingly narrow: GRPO+CoT adds only the plan prompt; PVE adds only the hierarchical (plan-level + code-level) advantage on top of that; APRA adds only posterior-UCB allocation and ω_i-weighting; PROSE adds only the two-stage schedule. No method-specific tuning of the optimiser, the KL coefficient, the batch size, or the decoding temperature was performed for any method, including ours. The three external checkpoints reported in Section 5 are evaluated under our evaluator and decoding protocol but were trained by their authors under their own pipelines; they are reference points, not budget- or training-matched comparisons, and their training hyperparameters are not ours to report.

Plan representation, embeddings, and clustering. No external embedding model is used anywhere in PROSE. In the forward setting, plan identity is given rather than inferred: the k plans are sampled explicitly and each rollout is generated conditioned on one of them, so plan membership is known by construction and no clustering step is required in the training loop. Where a vector representation of a plan is needed — as in the diversity audit of Appendix O.5 — it is the policy model's own final-layer hidden state at the `</plan>` token, so feature extraction adds no forward pass beyond standard rollout prompting and introduces no third-party model into the pipeline. Similarity is cosine,

Table 30: Optimiser, RL objective, reward, statistical reporting, and infrastructure. Every setting in this table is shared verbatim by all in-house methods; none of it was tuned per method, including for PROSE.

Hyperparameter	Value	Description and scope
Optimiser and RL objective		
Optimiser	AdamW	β_1=0.9, β_2=0.95, weight decay 0.01
Learning rate	1.0×10^{-6}	Constant (no schedule, no warm-up, no decay)
Global batch size	256	Per optimisation step, 8 GPUs, data parallel
Mini-batch size	128	Inner PPO-style update granularity
Micro-batch size	1	Gradient-accumulation granularity
β (KL)	1e−3	KL penalty coefficient against the frozen reference policy
ϵ_{clip}	0.2	GRPO importance-ratio clipping coefficient
Entropy coefficient	1e−3	Entropy bonus
Gradient clipping	1.0	Global gradient-norm clip
Reward		
Reward	Binary, $r \in \{0, 1\}$	Program passes the full test suite or not; no partial credit, no process reward
Evaluation and statistical reporting		
Seeds	$\{11, 23, 47\}$	Three independent runs for every in-house method
Significance test	Paired bootstrap	10,000 resamples over evaluation problems
TOST margin	$\delta = 1.0$	Pre-specified (not externally registered) equivalence margin in pass@1 points (vanilla GRPO, execution budget only)
Noise stress test	$p_{\text{noise}} \in \{0, 0.05, 0.10\}$	Reward-flip probability injected during RL post-training only
Infrastructure		
Hardware	8× NVIDIA H800 (80 GB)	Single node, data-parallel
Framework	VeRL + vLLM	vLLM as the rollout backend
Wall-clock per run	≈60 h	One 1,000-step run, Qwen2.5-Coder-Instruct-7B; PROSE adds +5.0% over GRPO and +1.8% over GRPO+CoT

thresholded at 0 to avoid negative pseudo-counts. Clustering is used in two places, both of them retrospective: the *retrospective* ICC measurement on unconditioned GRPO rollouts (Appendix K.3), which by construction emits no plan text, and the PVE-posthoc control of Appendix M, which reuses the same pipeline. In both, an auxiliary instruction-tuned LLM (Qwen2.5-Instruct-7B (Qwen Team, 2024), the same family as the policy) extracts a short algorithmic plan from each generated program, the extracted plans are embedded with the same hidden-state representation, and they are assigned to k=4 groups by greedy cosine assignment to centroids. The two uses differ in what happens next: the ICC measurement is purely observational and touches no training, whereas PVE-posthoc forms hierarchical advantages on the extracted clusters and trains on them for 1,000 steps. The auxiliary extractor is therefore never used by any method whose numbers appear in Table 2; it enters only that measurement and that diagnostic control.

Statistical protocol. All in-house numbers are means ± standard deviation over the three seeds $\{11, 23, 47\}$; three seeds is few, and we flag it as a limitation rather than lean on it. Significance for headline comparisons is assessed by paired bootstrap over evaluation problems with 10,000 resamples. The equivalence claim in the budget-scaling study, made only against vanilla GRPO and only in execution budget, uses a two one-sided tests (TOST) procedure with margin $\delta = 1.0$ pass@1 point. That margin was fixed a priori, as the smallest pass@1 difference we regard as practically relevant, before the N=8 versus N=32 comparison was examined; it was *not* registered on an external platform, and we therefore describe it as pre-specified rather than as externally registered.

Implementation cost. The incremental code over a VeRL GRPO baseline is ≈200 lines: the Beta–Bernoulli posterior update and the softmax-UCB score are closed-form on k=4 scalars per problem, plan grouping is bookkeeping over the sampled plan index, and the two-stage schedule is

Table 31: Per-method configuration for all in-house methods. The upper panel lists settings held fixed across every method; the lower panel lists, for each method, only what differs from that shared substrate. Decoding temperature appears in the shared panel and in no method's delta.

Shared by every in-house method (no method-specific override)	
Base model	Qwen2.5-Coder-Instruct-7B (Hui et al., 2024) by default; the 3B and DeepSeekCoder-7B (Guo et al., 2024) replications use the same protocol with the stated base model swapped in
Training data	APPS+ (Dou et al., 2024) (curated split with cleaned test suites)
Optimisation steps	1,000, from a common base checkpoint
Execution budget	$N = 16$ rollouts executed per input
Optimiser / LR	AdamW, 1e$-$6 constant, β_1=0.9, β_2=0.95, wd 0.01
Batch sizes	global 256 / mini 128 / micro 1
RL coefficients	KL $\beta = 1\text{e}{-3}$; clip ratio 0.2; entropy 1e$-$3; grad-norm clip 1.0
Decoding	$\tau_{\text{plan}} = \tau_{\text{code}} = 0.7$, top-$p = 1.0$, training and evaluation
Reward	Binary execution reward on the full test suite
Seeds / reporting	$\{11, 23, 47\}$; mean $\pm$ std over 3 seeds; paired bootstrap with 10,000 resamples
Hardware	8$\times$ H800 (80 GB), VeRL + vLLM
Method-specific deltas	
GRPO	None. Flat group-relative advantage $\widehat{A}_j = r_j - \bar{r}$ over N=16 i.i.d. rollouts; no plan prompt, no plan-level baseline, uniform (trivially, unstructured) budget use
GRPO + CoT	+ plan prompt prepended to the code prompt. Budget still uniform, advantage still flat; no plan-level baseline and no plan-conditioned grouping
PVE	+ k=4 sampled plans with balanced $M = N/k = 4$; hierarchical advantage (plan-level + code-level baselines); $\omega_i \equiv 1$ for the full run; no allocation machinery
APRA	+ posterior-UCB allocation from step 0 (no PVE warm-up): $M^{(0)}$=2, λ=1.0, η=0.05, τ_{alloc}=0.7, Beta(α_0, β_0)=(1, 1) prior, $\omega_i = M_i/(N/k)$
PROSE	+ the two-stage schedule on top of APRA: PVE for steps 0–500 (T_{PVE}=500), APRA for steps 500–1,000 (T_{APRA}=500)

a single configuration switch at step 500. A PVE-only recipe — hierarchical advantages with no allocation machinery at all — is ≈100 lines and captures +3.2 of the +5.3 total gain (60%), which is the configuration we would suggest to an adopter who wants most of the benefit for a minimal change. The runtime cost of the added machinery is negligible against LLM inference: the measured wall-clock overhead of the full method is +5.0% over vanilla GRPO and +1.8% over GRPO+CoT for a 1,000-step run on identical hardware, essentially all of it attributable to generating the 35–40 extra plan tokens per problem rather than to the posterior or allocation arithmetic.

S PROMPT TEMPLATES

Every PROSE rollout is generated in two steps, and this is true in both training stages: a plan is first sampled from the plan-generation template, then a program is sampled from the code-generation template with that plan injected as explicit context. The two templates below are therefore *functional* (one produces plans, one produces code) rather than stage-specific. PVE and APRA differ in how the N executions are distributed over the k plans and in how the resulting advantages are weighted; they do not differ in which prompt is used, and both templates are in use throughout both stages.

The plan-generation template constrains the model to a single natural-language sentence inside a fixed XML-like wrapper (`<plan>...</plan>`) and explicitly forbids code, pseudocode, and variable-level implementation detail. The constraint does real work in three places. It fixes plan granularity at the coarse-to-mid level the latent-plan model of Section 3.1 assumes, so that plan identity tracks algorithmic strategy rather than surface phrasing. It makes the `</plan>` token a stable, well-defined position at which to read the hidden-state plan embedding used by the diversity audit (Appendix O.5), and would be the natural position for the unrun contextual-posterior extension of Appendix B. And it bounds the token overhead of the method: the sampled plans measure 35–40 tokens per problem, which is why PROSE's planning-token cost is a fraction of an unconstrained chain-of-thought preamble's: in the Case-1 matched-success study of Appendix O.9, 153 planning

tokens for PROSE against 612 for GRPO+CoT on that problem, with PROSE using fewer generated tokens in total despite producing four distinct plans.

Prompt fairness. For in-house baselines we align the system and user templates with PROSE as closely as the method definitions allow. GRPO+CoT keeps the identical plan-then-code prompt structure and differs only in disabling PVE/APRA and allocating rollouts uniformly, so the plan prompt itself is not a PROSE-only advantage and its contribution is isolated as the GRPO $\rightarrow$ GRPO+CoT step of the attribution ladder in Section 5. For methods with no explicit planning step the plan slot is omitted while the remaining instructions stay aligned. External baselines with officially released prompt formats are evaluated following their authors' recommended settings, under our evaluator and the shared decoding protocol of Table 29. Below we show both templates instantiated on a single problem.

Plan Generation Prompt.

```
<|im_start|>system
Output exactly ONE natural language sentence plan wrapped exactly as:
<plan>one short sentence</plan>
Do NOT output code, pseudocode, bullet points, or line breaks.
Do NOT include variable-level implementation details.
You MUST include the closing tag </plan>.
<|im_end|>

<|im_start|>user
Write a python function to check whether the length of the word is odd or
     not. The function head is: def word_len(s):
<|im_end|>

<|im_start|>assistant
Your Response:
```

Code Generation Prompt.

```
<|im_start|>system
Write a function to solve the problem.
<|im_end|>

<|im_start|>user
Task: Write a python function to check whether the length of the word is
    odd or not. The function head is: def word_len(s):

Plan: Check whether the length of the input string is odd by applying the
     modulus operator to test divisibility by two, then return a boolean
    indicating whether the remainder is nonzero.
<|im_end|>

<|im_start|>assistant
Your Response:
```