LURE: PURSUIT-EVASION SELF-PLAY FOR ZERO-DATA MULTI-STEP REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards (RLVR) has substantially improved large language model reasoning, but its scalability remains tied to externally supplied tasks whose difficulty must keep pace with an improving policy. Zero-data self-play addresses this bottleneck by generating its own curriculum, yet existing approaches either estimate task difficulty from model-derived proxies or rely on executable task specifications, while multi-step solvers remain largely trained from sparse outcome supervision. We introduce LURE, a verifier-grounded pursuit–evasion framework for zero-data self-play, in which a task-generating Challenger and a planner–executor Solver co-evolve through a shared environment verifier. This formulation turns the verifier into a common learning interface for both curriculum construction and policy improvement. Specifically, LURE uses verifier-confirmed Solver success to position generated tasks near a *capture frontier*, and converts stepwise verifier progress into *capture-anchored dense process credit* for the Solver. Round-anchored regularization further stabilizes the two adaptive policies during co-evolution. Across three heterogeneous reasoning environments and three backbone families, LURE outperforms advanced baselines under both unified and specialist settings, and across out-of-domain benchmarks it delivers the strongest near-domain transfer and the best overall average.

1 INTRODUCTION

Large language models are increasingly trained on verifiable reasoning tasks that require coordinated multi-step decision making, including multi-hop question answering (Gong et al., 2025), constrained instruction following (Zhou et al., 2023), and logic-grid deduction (Lin et al., 2025). A prominent route to such capabilities is reinforcement learning with verifiable rewards (RLVR), which optimizes the policy against programmatic or environment-based correctness signals rather than a learned reward model (Guo et al., 2025). However, its scalability depends on a supply of externally constructed tasks with reliable verification, whose coverage and difficulty must keep pace with the improving policy (Fang et al., 2026). Zero-data self-evolving training offers a way past this bottleneck: rather than relying on a fixed externally supplied task set, the model generates its own curriculum and improves from automatically verifiable feedback (Huang et al., 2025a; Zhao et al., 2026a).

Existing zero-data self-evolving methods derive their training signal from two main sources (as shown in **Fig. 1**a,b). *Pseudo-label* methods construct supervision from model-generated outputs, typically using majority voting as a proxy for correctness (Zuo et al., 2026), while self-play variants further use agreement among Solver rollouts to estimate the difficulty of generated tasks (Huang et al., 2025a). *Executor-grounded* methods instead derive feedback from executable specifications, such as program execution, verification functions, or unit tests (Zhao et al., 2026a; Zhou et al., 2026). Both families demonstrate that models can bootstrap useful training signals without human-labeled task data, but neither defines a general solver-relative frontier from verifier-confirmed success in the target environment. Rollout agreement need not coincide with correctness, while executor grounding requires tasks to admit an executable or testable specification. A general zero-data curriculum therefore requires a solver-relative and verifier-grounded signal that positions generated tasks using the same environment verifier that ultimately defines task success.

A second limitation arises on the Solver side. In multi-step self-play, outcome verification provides only a terminal reward, offering no direct credit for the intermediate actions that advance the so-

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

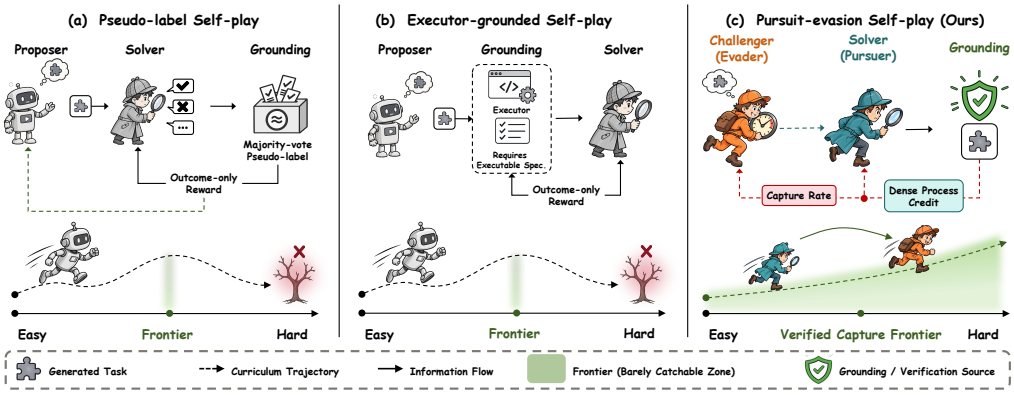


Figure 1: Comparison of zero-data self-play paradigms. (a) Pseudo-label self-play grounds rewards in Solver agreement (Huang et al., 2025a). (b) Executor-grounded self-play requires executable task specifications (Zhao et al., 2026a). Neither targets a verified Solver-relative frontier. (c) LURE derives the Challenger's capture rate and the Solver's dense process credit from one shared verifier.

lution. Under group-relative objectives, the learning signal further vanishes when all rollouts in a group receive the same outcome, with all-failure groups becoming increasingly likely as generated tasks move beyond the Solver's frontier (Prakash & Buvanesh, 2025). Process reward models can densify supervision, but conventionally require additional process annotations or a separately trained evaluator (Lightman et al., 2024), weakening the zero-data premise. Meanwhile, co-training the task generator and Solver continually shifts the training distribution, creating a moving-target optimization problem that can destabilize co-evolution. A fully self-contained loop therefore requires verifier-derived step-level credit and explicit stabilization of the two co-evolving policies.

To address these, we propose LURE, a verifier-grounded pursuit–evasion framework for zero-data self-play (as shown in **Fig. 1**c). An LLM Challenger acts as the evader by generating tasks near the current Solver's *verified capture frontier*, while a planner–executor Solver acts as the pursuer by solving them through multi-step interaction. The central design is to use the same environment verifier for both sides of this game: verifier-confirmed capture statistics provide the Challenger with a solver-relative curriculum signal, while changes in verifier-observable progress provide the Solver with *capture-anchored dense process credit*. This coupling turns verification from a terminal acceptance test into a shared learning signal for task generation and multi-step reasoning. To keep the two adaptive policies from drifting apart during co-training, LURE further anchors their updates within each self-play round. Together, these components keep the curriculum and the Solver advancing in step, so that generated tasks stay both reachable and informative.

Contributions. **(1)** We formulate zero-data self-play for verifiable multi-step reasoning as a pursuit–evasion problem, in which a task-generating Challenger and a planner–executor Solver co-evolve through feedback from a shared environment verifier. **(2)** We introduce LURE, a verifier-grounded co-evolution framework that couples adaptive curriculum construction with multi-step policy improvement, using the same verifier as the learning interface for both players. **(3)** LURE realizes this coupling through two complementary signals: a *capture-frontier* objective that trains the Challenger from verifier-confirmed Solver success, and *capture-anchored dense process credit* that trains the Solver from stepwise verifier progress, with round-anchored regularization stabilizing their joint evolution. **(4)** Across three heterogeneous reasoning environments and three backbone families, LURE outperforms advanced zero-data and curated-task baselines under both unified and specialist settings, and it achieves the best average across thirteen OOD benchmarks.

2 Related Work

Verifiable Multi-Step Reasoning. Recent progress in multi-step reasoning increasingly combines explicit intermediate computation with verifiable supervision. Chain-of-thought exposes intermediate reasoning trajectories (Wei et al., 2022), while process supervision and process reward models provide step-level feedback beyond final-answer correctness (Lightman et al., 2024; Wang et al., 2024a). Reinforcement learning with verifiable rewards (RLVR) instead converts programmatic

correctness signals into direct policy optimization objectives, with group-relative methods widely adopted for reasoning LLMs (Shao et al., 2024; Guo et al., 2025; Sheng et al., 2024). In interactive settings, agents further alternate decomposition, grounded action, observation, and revision over multiple steps (Yao et al., 2022; Shinn et al., 2023). Programmatically verifiable benchmarks provide corresponding testbeds for constrained instruction following (Zhou et al., 2023), logic-grid deduction (Lin et al., 2025), and multi-hop relational question answering (Gong et al., 2025). Despite substantial progress in optimizing multi-step solvers, most of this literature treats the task distribution as fixed or externally supplied rather than as part of the learning process.

Zero-Data Self-Evolving Reasoning. Zero-data self-evolving methods improve training signals from the model itself or automatically verifiable environments. *Pseudo-label* approaches derive supervision from model-generated signals such as majority voting, confidence, or entropy (Zuo et al., 2026; Zhao et al., 2026b; Zhang et al., 2026a), whereas *executor-grounded* approaches obtain feedback from program execution, unit tests, tool outcomes, or formal verification (Dong et al., 2025; Zhou et al., 2026; Xia et al., 2025). Building on these signals, proposer–solver systems adapt task generation to the current learner: Absolute Zero uses execution-grounded learnability (Zhao et al., 2026a), SQLM favors tasks of intermediate solvability under majority voting (Zhang et al., 2026b), and R-Zero trains its Challenger from Solver agreement and uncertainty with informative-band selection (Huang et al., 2025a). Learner-relative task generation has precedents in asymmetric self-play, intermediate-success goal generation, and regret-based environment design (Sukhbaatar et al., 2017; Florensa et al., 2017; Dennis et al., 2020). These works establish the value of adaptive curricula, but curriculum construction and Solver optimization are typically addressed with distinct signals, and outcome-only group-relative training can lose its learning signal when all rollouts receive the same reward (Prakash & Buvanesh, 2025). LURE instead uses a shared target-environment verifier to ground both learner-relative task placement and step-level Solver credit.

3 METHODOLOGY

Problem Formulation. We consider verifiable language reasoning environments $\mathcal{E}$. Each environment $e \in \mathcal{E}$ provides a task generator $x = g_e(d, \omega)$, where $d \in [0, 1]$ is an environment-defined difficulty parameter and ω denotes generation randomness, together with a multi-turn interaction protocol and a programmatic verifier. For a Solver trajectory τ of T_τ turns, the verifier returns a terminal *capture* signal $R(\tau) \in \{0, 1\}$ and a best-so-far verified progress $\phi_t \in [0, 1]$, with $\phi_0 = 0$ and $\phi_{T_\tau} = 1$ upon capture. We define $\Delta\phi_t = \phi_t - \phi_{t-1} \geq 0$. We formulate zero-data self-play as a pursuit–evasion game. A Challenger $\pi_C(d \mid e)$, the evader, selects task difficulty, while a planner–executor Solver $\pi_S = (\pi_{\text{plan}}, \pi_{\text{exec}})$, the pursuer, solves the resulting task. For G Solver rollouts on task x, we define its empirical capture rate as:

$$\hat{p}_S(x) = \frac{1}{G} \sum_{g=1}^{G} \mathbb{1}[R(\tau_g) = 1], \qquad \tau_g \sim \pi_S(\cdot \mid x), \tag{1}$$

which provides a learner-relative measure of task difficulty. We write $p(x)$ for $\hat{p}_S(x)$ when the Solver is clear from context. With $e \sim \mu$ and $\omega \sim \nu_e$, the coupled objectives are

$$\max_{\pi_C} \quad J_C(\pi_C; \pi_S) = \mathbb{E}_{e\sim\mu,\, d\sim\pi_C(\cdot|e),\, \omega\sim\nu_e} \left[r_C(g_e(d, \omega)) \right], \tag{2}$$

$$\max_{\pi_S} \quad J_S(\pi_S; \pi_C) = \mathbb{E}_{e\sim\mu,\, d\sim\pi_C(\cdot|e),\, \omega\sim\nu_e,\, \tau\sim\pi_S(\cdot|g_e(d,\omega))} \left[r_S(\tau; R, \phi) \right]. \tag{3}$$

We instantiate r_C and r_S below as the *capture-frontier* objective and *capture-anchored dense process credit*, respectively. Both are built on verifier outcomes and require no human-curated training tasks, process annotations, or a learned reward model.

Overview. Figure 2 illustrates the training loop of LURE. The Challenger first selects task difficulties, after which the environment generator instantiates the corresponding tasks. The planner–executor Solver then produces multi-turn rollouts, while the verifier records terminal capture and stepwise verified progress. These signals close the self-play loop in complementary ways: capture statistics adapt the Challenger toward the Solver's current frontier, stepwise progress provides dense credit to the executor, and the progress realized by each plan supervises the planner. Round-anchored regularization stabilizes all three policies as they co-evolve.

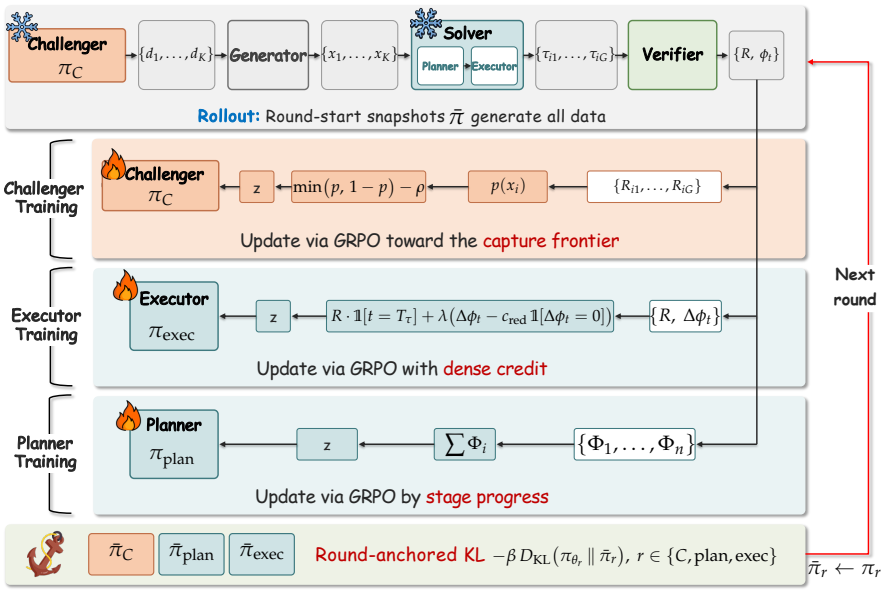


Figure 2: Overview of LURE. Round-start snapshots generate all rollout data. Verifier-confirmed capture statistics train the Challenger toward the capture frontier, while terminal capture and step-wise progress provide dense credit to the executor and realized stage progress trains the planner. Round-anchored KL stabilizes all three policies before the next self-play round.

Capture-Frontier Challenger. The Challenger learns where to place tasks relative to the current Solver rather than generating their content directly. For each sampled difficulty $d \sim \pi_C(\cdot \mid e)$, the environment instantiates $x = g_e(d, \omega)$, whose G Solver rollouts yield the empirical capture rate $p(x)$ in Eq. (1). We define the Challenger reward as

$$r_C(x) = \min\big(p(x),\, 1 - p(x)\big) - \rho(x), \tag{4}$$

where $\rho(x)$ is a diversity penalty described below. The frontier term peaks at $p(x) = \frac{1}{2}$, favoring tasks the Solver neither always fails nor always solves. To prevent the Challenger from concentrating on a narrow family of tasks, we use

$$\rho(x) = \frac{n_{\text{sig}}(x) - 1}{N}, \tag{5}$$

where $n_{\text{sig}}(x)$ counts the N proposals of environment e in the round that share a coarse signature with x. The rewards are group-normalized for the Challenger's GRPO update. Each prompt also carries a per-slot difficulty hint and a competence summary ζ_e over difficulty levels, allowing the frontier to track the evolving Solver. We suppress both in $\pi_C(d \mid e)$ for brevity.

Proposition 1 (Frontier maximizes the group-relative signal)**.** *Let the $G \geq 2$ capture outcomes of a task be independent* Bernoulli(q). *The outcome-only group-normalized advantages are nonzero if and only if the outcomes are not all equal, an event of probability $\psi_G(q) = 1 - q^G - (1-q)^G$. Both $\psi_G(q)$ and the frontier term* $\min(q, 1-q)$ *are uniquely maximized at $q = \frac{1}{2}$.*

The frontier term thus peaks where the Solver is most likely to receive a non-degenerate group-relative signal, and in expectation this maximizer is unchanged when q is replaced by its finite-G estimate. **App. D** proves this result and the properties stated below.

Capture-Anchored Dense Solver Credit. Outcome-only supervision becomes uninformative when all G rollouts of a task share the same terminal result, particularly in the all-failure regime beyond the Solver's frontier. We therefore derive step-level Solver credit directly from verified progress. For executor turn t in trajectory τ, we define

$$r_t = R(\tau)\,\mathbb{1}[t = T_\tau] + \lambda\,(\Delta\phi_t - c_{\text{red}}\mathbb{1}[\Delta\phi_t = 0])\,, \tag{6}$$

where the terminal capture is assigned only to the final turn and c_{red} penalizes turns that make no verified progress. We convert these raw rewards into turn-level advantages by jointly normalizing all executor turns from the G rollouts of task x as $A_t = z_{\mathcal{G}(x)}(r_t)$, where $\mathcal{G}(x)$ denotes this task-conditioned turn group. This construction preserves terminal success as an anchor while exposing within-trajectory variation that remains informative even when every rollout fails. Group normalization cancels any unconditional per-turn cost exactly (App. D, Prop. 4), whereas the conditional cost can still reweight stalled turns. Moreover, because ϕ_t is best-so-far,

$$\sum_{t=1}^{T_\tau} \Delta\phi_t = \phi_{T_\tau}, \tag{7}$$

so verified progress accumulates monotonically and later regressions cannot erase progress already established by the verifier. This shaping remains capture-preserving: with $T_{\max}$ the maximum number of executor turns, if $\lambda c_{\mathrm{red}}(T_{\max} - 1) < 1$, every capturing trajectory receives strictly larger total raw credit than any non-capturing trajectory, and assigning $R(\tau)$ only at termination avoids the positive length bias induced by replicating capture across turns. Our configuration satisfies this condition with $\lambda c_{\mathrm{red}}(T_{\max} - 1) = 0.2375$ (App. D, Cor. 2). The planner is credited by the progress realized during the executor segments induced by its decisions. Let segment i span turns $[a_i, b_i]$ and define its realized progress:

$$\Phi_i = \phi_{b_i} - \phi_{a_i - 1}. \tag{8}$$

We score the corresponding planner rollout by

$$r^{\mathrm{plan}}(\tau) = \sum_i \Phi_i, \tag{9}$$

and normalize these scores within the same task group. Since the segments tile the trajectory, this score equals the peak verified progress ϕ_{T_τ}. Thus, executor updates receive local step-level credit, whereas planner updates reflect the verified progress their induced executions achieve.

Round-Anchored Co-Evolution. Co-evolution is inherently non-stationary: updates to the Challenger change the Solver's task distribution, while updates to the Solver change the capture statistics that supervise the Challenger. At the beginning of each round, we freeze snapshots $\bar{\pi}_C$, $\bar{\pi}_{\mathrm{plan}}$, and $\bar{\pi}_{\mathrm{exec}}$, which generate all data in that round. Each role is then optimized on this fixed batch with a KL penalty to its own round-start snapshot. These snapshots therefore serve as behavior policies and proximal anchors, limiting within-round drift while keeping each reference aligned with the data-generating policy. The updated policies interact only after the anchors are refreshed for the next round, yielding a simultaneous-move co-evolution process. At round start, the behavior and reference policies coincide, so $\nabla_\theta D_{\mathrm{KL}}(\pi_\theta \,\|\, \bar{\pi})\big|_{\theta=\bar{\theta}} = 0$, leaving the first-order group-relative update unchanged (App. D, Prop. 3). A stale reference generally does not satisfy this property.

Unified GRPO Optimization Objective. The Challenger, planner, and executor share the same clipped GRPO objective (Shao et al., 2024; Schulman et al., 2017) and differ only in how their advantages are constructed. For role $r \in \{C, \mathrm{plan}, \mathrm{exec}\}$, let $\bar{\pi}_r$ denote its round-start snapshot and define the token-level policy ratio as follows:

$$\varrho_{r,j} = \frac{\pi_{\theta_r}(y_j \mid s_j)}{\bar{\pi}_r(y_j \mid s_j)}. \tag{10}$$

Each role maximizes

$$\mathcal{J}_r(\theta_r) = \mathbb{E}_j\left[\min\left(\varrho_{r,j} A_j^r,\ \mathrm{clip}(\varrho_{r,j}, 1-\epsilon, 1+\epsilon) A_j^r\right) - \beta\, D_{\mathrm{KL}}(\pi_{\theta_r}(\cdot \mid s_j) \,\|\, \bar{\pi}_r(\cdot \mid s_j))\right]. \tag{11}$$

The role-specific advantages are obtained from the rewards defined above:

$$A^C = z_{\mathrm{slot}}(r_C), \qquad A_t^{\mathrm{exec}} = z_{\mathcal{G}(x)}(r_t), \qquad A_\tau^{\mathrm{plan}} = z_{\mathcal{T}(x)}\left(r^{\mathrm{plan}}(\tau)\right), \tag{12}$$

where slot collects the Challenger samples drawn from the same prompt and $\mathcal{T}(x) = \{\tau_g\}_{g=1}^G$ denotes the rollout-level group for task x. Each generated token inherits the advantage of the Challenger sample, executor turn, or planner rollout to which it belongs. Alg. 1 shows the training round.

Algorithm 1 One round of LURE. Full workflow is provided in **Alg. 2**.

Require: policies $\pi_C, \pi_{\text{plan}}, \pi_{\text{exec}}$, environments $\mathcal{E}$, candidates K, rollouts G
Ensure: updated policies $\pi_C, \pi_{\text{plan}}, \pi_{\text{exec}}$
round-start snapshots $\bar{\pi}_r \leftarrow \pi_r$ for $r \in \{C, \text{plan}, \text{exec}\}$, buffers $\mathcal{D}_C, \mathcal{D}_{\text{plan}}, \mathcal{D}_{\text{exec}} \leftarrow \emptyset$, task pool $\mathcal{X} \leftarrow \emptyset$
for each environment $e \in \mathcal{E}$ **do**
    draw K difficulties $d \sim \bar{\pi}_C(\cdot \mid e)$ and add each valid $x = g_e(d, \omega)$ to $\mathcal{X}$
end for
for each task $x \in \mathcal{X}$ **do**
    roll out G trajectories with $(\bar{\pi}_{\text{plan}}, \bar{\pi}_{\text{exec}})$, and the verifier returns $R(\tau_g)$ and $\{\phi_t\}$
    compute $p(x)$ (Eq. 1) and add the Challenger sample with $r_C(x)$ (Eq. 4) to $\mathcal{D}_C$
    add the executor turns with r_t (Eq. 6) to $\mathcal{D}_{\text{exec}}$
    add the planner rollouts with $r^{\text{plan}}(\tau_g)$ (Eq. 9) to $\mathcal{D}_{\text{plan}}$
end for
group-normalize the rewards into A^C, A_t^{exec}, A_τ^{plan} (Eq. 12)
for each role $r \in \{C, \text{plan}, \text{exec}\}$ **do**
    update π_r on $\mathcal{D}_r$ by maximizing $\mathcal{J}_r$ (Eq. 11) with KL anchor $\bar{\pi}_r$
end for
return $\pi_C, \pi_{\text{plan}}, \pi_{\text{exec}}$

4 EXPERIMENTS

Benchmarks & Metrics. We evaluate LURE on three programmatically verifiable environments spanning distinct forms of multi-step reasoning: **PhantomWiki** (Gong et al., 2025) for multi-hop relational inference, **IFEval** (Zhou et al., 2023) for constrained instruction following, and **ZebraLogic** (Lin et al., 2025) for logic-grid deduction. Each provides an automatic verifier for self-play, with evaluation on held-out instances. PhantomWiki reports chain-verified exact match on 120 questions, requiring all reasoning hops to be correct. IFEval reports strict per-prompt accuracy on 541 prompts. ZebraLogic reports puzzle and cell accuracy on 300 puzzles stratified by official grid size, with puzzle accuracy requiring a fully correct grid. We further evaluate general reasoning on **MMLU-Pro** (Wang et al., 2024b), **SuperGPQA** (Du et al., 2026), and **BBH** (Suzgun et al., 2023). Full protocols and metric definitions are in App. B.

Models & Baselines. We evaluate three instruction-tuned backbones: Qwen2.5-7B-Instruct, Llama-3.1-8B-Instruct, and Gemma-2-9B-it. We compare LURE with four baselines. *Base* is the backbone evaluated zero-shot. *GRPO* (Shao et al., 2024) is trained on externally curated tasks. *GRPO-Zero* is trained on a self-generated pool collected in the first round and then fixed, which isolates curriculum adaptation from self-generated data alone. *R-Zero* (Huang et al., 2025a) is an adaptive zero-data proposer–solver method guided by Solver agreement and uncertainty. For each backbone, all methods share the same evaluation verifier and protocol. Every method is trained either as a unified model, with one set of weights on all three environments, or as per-environment specialists. Baseline constructions are in App. A.

Implementation Details. We implement all trainable methods with verl-GRPO (Sheng et al., 2024; Shao et al., 2024) under FSDP-2 on 16× RTX 6000 Pro 96 GB GPUs. LURE is trained for eight self-play rounds. Each round generates 48 candidate tasks per environment (8 prompt slots ×6 samples), with each task evaluated by $G = 8$ Solver rollouts. We set the dense-credit weight to $\lambda = 0.25$, the zero-progress cost to $c_{\text{red}} = 0.05$, and use round-anchored KL with $\beta = 0.1$, a learning rate of 5×10^{-6}, and one PPO epoch per round for all three policies. Baselines share the settings in Table 7. Full hyperparameters and prompts are in App. A.

4.1 MAIN RESULTS

Table 1 reports unified and specialist results across three backbones. LURE achieves the best multi-step reasoning average in every setting and the highest overall average for each backbone. Gains are largest on PhantomWiki: with Qwen2.5-7B, the unified model exceeds R-Zero by 15.0 points and GRPO-Zero by 28.3, while the specialist reaches 80.0% success. The gain of R-Zero over GRPO-Zero shows the value of continual adaptation, and the further gain of LURE supports grounding task placement in verifier-confirmed Solver success. LURE also leads trained methods on IFEval

and both ZebraLogic metrics for unified models. Its general-reasoning average likewise exceeds the corresponding Base average for every backbone under both unified and specialist training.

Table 1: Main results (%) across three backbones. *Average* denotes the mean within each category, using ZebraLogic puzzle accuracy for the multi-step aggregate. *Overall Avg.* averages the multi-step and general-reasoning scores. Specialist general-reasoning scores are averaged over the three environment-specific models. Best and second-best results: boxed and underlined.

		Qwen2.5-7B-Instruct (Qwen et al., 2025)					**Llama-3.1-8B-Instruct** (Grattafiori et al., 2024)					**Gemma-2-9B-it** (Team et al., 2024)				
	Benchmark	**Base**	**GRPO**	**GRPO-Zero**	**R-Zero**	**LURE (Ours)**	**Base**	**GRPO**	**GRPO-Zero**	**R-Zero**	**LURE (Ours)**	**Base**	**GRPO**	**GRPO-Zero**	**R-Zero**	**LURE (Ours)**
Unified Model	*Multi-Step Reasoning*															
	PhantomWiki (Succ.)	25.0	43.3	37.5	50.8	**65.8**	0.0	0.8	0.8	3.3	**4.2**	11.7	27.5	15.0	34.2	**40.0**
	IFEval (Prompt)	**63.0**	60.6	58.4	57.9	61.2	37.5	38.8	39.4	42.5	**44.5**	60.4	64.0	61.2	62.5	**66.2**
	ZebraLogic (Puzzle)	11.6	15.4	13.9	14.2	**16.9**	12.8	14.1	13.0	14.6	**15.9**	13.1	15.7	14.3	15.1	**18.2**
	ZebraLogic (Cell)	42.3	45.8	44.2	43.6	**47.2**	13.7	27.9	26.4	26.8	**29.5**	38.4	40.2	39.9	41.6	**43.5**
	Average	33.2	39.8	36.6	41.0	**48.0**	16.8	17.9	17.7	20.1	**21.5**	28.4	35.7	30.2	37.3	**41.5**
	General Reasoning															
	MMLU-Pro	56.4	55.0	54.7	**58.3**	58.2	47.9	**50.1**	47.7	50.0	49.4	49.8	49.6	49.6	50.9	**52.5**
	SuperGPQA	29.1	28.7	29.0	**31.5**	31.3	23.0	23.4	22.2	**24.7**	22.7	22.5	22.1	22.3	23.3	**25.8**
	BBH	72.6	70.2	72.0	**74.7**	**74.7**	62.3	62.7	60.7	62.2	**63.1**	52.2	50.6	50.4	**53.6**	53.5
	Average	52.7	51.3	51.9	**54.8**	54.7	44.4	45.4	43.5	**45.6**	45.1	41.5	40.8	40.8	42.6	**43.9**
	Overall Avg.	42.9	45.5	44.2	47.9	**51.4**	30.6	31.6	30.6	32.9	**33.3**	35.0	38.2	35.5	40.0	**42.7**
Per-environment Specialists	*Multi-Step Reasoning*															
	PhantomWiki (Succ.)	25.0	60.8	0.0	56.7	**80.0**	0.0	2.5	0.8	4.2	**5.0**	11.7	40.0	20.8	30.0	**42.5**
	IFEval (Prompt)	**63.0**	62.8	54.9	60.1	**63.0**	37.5	43.4	39.4	43.1	**44.7**	60.4	62.5	60.4	63.6	**66.2**
	ZebraLogic (Puzzle)	11.6	16.1	12.7	15.0	**18.3**	12.8	13.6	12.9	13.6	**15.2**	13.1	15.7	14.3	15.0	**17.6**
	ZebraLogic (Cell)	42.3	46.5	43.1	44.8	**48.6**	13.7	27.2	26.2	28.3	**29.9**	38.4	40.3	39.6	40.9	**43.1**
	Average	33.2	46.6	22.5	43.9	**53.8**	16.8	19.8	17.7	20.3	**21.6**	28.4	39.4	31.8	36.2	**42.1**
	General Reasoning															
	MMLU-Pro	56.4	57.9	55.8	55.6	**58.2**	**47.9**	47.4	47.8	47.3	47.6	49.8	51.0	48.9	**51.7**	49.4
	SuperGPQA	29.1	28.8	28.1	**31.4**	30.6	23.0	22.3	19.9	**24.0**	22.4	22.5	21.5	19.3	**23.7**	23.6
	BBH	72.6	73.2	72.8	73.5	**74.3**	62.3	61.3	60.7	61.6	**63.7**	52.2	51.3	50.0	52.0	**53.5**
	Average	52.7	53.3	52.2	53.5	**54.4**	44.4	43.7	42.8	44.3	**44.6**	41.5	41.3	39.4	**42.5**	42.2
	Overall Avg.	42.9	50.0	37.4	48.7	**54.1**	30.6	31.7	30.2	32.3	**33.1**	35.0	40.3	35.6	39.3	**42.1**

4.2 GENERALIZATION AND TRANSFER

Difficulty Extrapolation. We extend each generator beyond its hardest training level $d = 1$ to $d \in \{1.25, 1.5\}$, corresponding to 7–8 hop PhantomWiki questions, 6–7 constraint IFEval prompts, and 7×7–8×8 ZebraLogic grids. Table 2 shows that LURE achieves the best result in five of the six extrapolated settings. The advantage is largest on PhantomWiki, where LURE reaches 19.0% at 8 hops, compared with 9.0% for the strongest baseline. The gains indicate that the learned curriculum improves robustness beyond the difficulty range observed during training.

Table 2: Difficulty extrapolation (%) from d=1 to d=1.5 on the Qwen2.5-7B-Instruct unified model.

d	**Structure**	**Base**	**GRPO**	**GRPO-Zero**	**R-Zero**	**LURE (Ours)**
PhantomWiki (Succ.)						
1.0	6 hops	10.0	25.0	17.0	33.0	**51.0**
1.25	7 hops	7.0	13.0	5.0	14.0	**23.0**
1.5	8 hops	1.0	9.0	4.0	6.0	**19.0**
IFEval (Prompt)						
1.0	k=5	**53.0**	39.0	46.0	38.0	46.0
1.25	k=6	**36.0**	32.0	33.0	25.0	31.0
1.5	k=7	31.0	30.0	19.0	21.0	**35.0**
ZebraLogic (Cell)						
1.0	6×6	33.6	32.5	34.4	32.2	**38.8**
1.25	7×7	27.6	25.2	23.6	29.7	**31.1**
1.5	8×8	22.5	22.4	21.8	22.3	**25.8**

Cross-Environment Transfer. Figure 3 evaluates each per-environment specialist on the other two environments, reporting gains over the untrained backbone. LURE shows the most consistent positive transfer. Its ZebraLogic specialist improves PhantomWiki by 14.2 points, compared with 2.5 for GRPO and no gain for R-Zero, while its IFEval specialist also improves both other environments. These results indicate that LURE learns transferable reasoning behaviors beyond the environment used for training, rather than environment-specific shortcuts.

Out-of-Domain Benchmarks. Table 3 separates near-domain benchmarks that share a skill with one training environment, namely CLUTRR (Sinha et al., 2019), LiveBench (White et al., 2024), and IFBench (Pyatkin et al., 2026), from far-transfer benchmarks spanning mathematics (MATH500 (Hendrycks et al., 2021; Lightman et al., 2024), AIME (Dekoninck et al., 2026), OlympiadBench (He et al., 2024)), logic (KOR-Bench (Ma et al., 2025), BBEH (Kazemi et al., 2025)), and other reasoning (StepGame (Shi et al., 2022), MuSR (Sprague et al., 2024)), with protocols in App. B. LURE achieves the strongest near-domain average at 38.2, compared with 36.9

for the closest baseline, and the best overall average of 36.9 across all thirteen benchmarks. Its advantage extends to the far-transfer logic and other-reasoning groups.

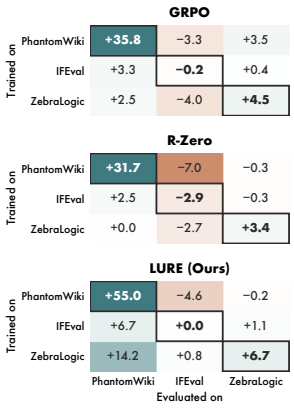


Figure 3: Specialist transfer across environments, in points against the untrained backbone. Boxed diagonal cells are in-domain. Full scores are in **Table 10**.

Table 3: OOD generalization (%) of the Qwen2.5-7B-Instruct unified model with benchmark-specific prompts and no planner or executor role. Math benchmarks report Avg@16 and *Overall* is the mean over all benchmarks. Best and second-best results: boxed and underlined.

	Benchmark	Metric	Base	GRPO	GRPO-Zero	R-Zero	LURE (Ours)
Near-Domain	CLUTRR	Match	25.3	26.6	21.3	28.0	**31.6**
	LiveBench-IF	Score	52.1	49.3	51.1	51.2	**56.1**
	IFBench	Prompt	31.3	29.3	32.0	**38.3**	31.0
	LiveBench-Reasoning	Score	28.4	29.0	27.2	30.0	**34.0**
	Average		34.3	33.6	32.9	36.9	**38.2**
Far Transfer	*Mathematics*						
	MATH500	Avg@16	75.7	73.3	71.8	**77.1**	77.0
	AIME24	Avg@16	13.8	12.1	10.6	**16.0**	14.8
	AIME25	Avg@16	6.7	1.9	7.3	**11.0**	8.5
	AMC23	Avg@16	53.4	58.9	50.0	**59.5**	56.6
	OlympiadBench	Avg@16	39.2	37.3	38.2	37.9	**41.3**
	Average		37.8	36.7	35.6	**40.3**	39.6
	Logic						
	KOR-Bench	Acc.	38.6	38.0	37.7	**42.6**	**42.6**
	BBEH	Micro	**12.5**	11.4	11.9	10.5	12.1
	Average		25.6	24.7	24.8	26.6	**27.4**
	Other Reasoning						
	StepGame	Acc.	15.0	13.2	15.1	17.7	**18.2**
	MuSR	Acc.	50.4	50.3	45.6	53.8	**55.3**
	Average		32.7	31.8	30.4	35.8	**36.8**
	Overall Avg.		34.0	33.1	32.3	36.4	**36.9**

4.3 Framework Analysis

Ablation Study. Figure 4 removes each curriculum, Solver-credit, and stabilization component of LURE in turn. All removals reduce performance across the three environments. The largest degradation comes from removing the learned Challenger, while the capture-frontier reward and repetition penalty also contribute consistently. Dense credit, the zero-progress cost, and planner credit each contribute 2.2 to 4.5 points, and both stabilization ablations cause substantial drops. No single component accounts for the full gain. Full results are in Table 11.

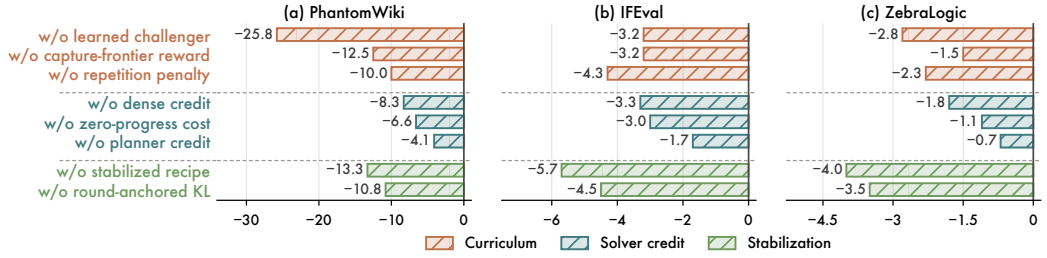


Figure 4: Ablation of the Qwen2.5-7B-Instruct unified model.

Hyperparameter Sensitivity. Figure 5 varies six hyperparameters around their defaults. Performance remains stable near the selected values, whereas extreme settings degrade both PhantomWiki and IFEval. The largest drops occur without round anchoring ($\beta = 0$), with a small rollout group ($G = 2$), or without the repetition penalty. The sensitivity to G agrees with Proposition 1: smaller groups are more likely to share the same terminal outcome and provide no group-relative signal. The dense-credit weight λ and zero-progress cost c_{red} show interior optima, while gains saturate beyond $K = 48$. The defaults lie in a broad high-performing region.

Curriculum Dynamics. Figure 6 traces curriculum adaptation on PhantomWiki. Starting well below the target capture frontier of $p = 0.5$, LURE reduces task difficulty until the capture rate approaches the frontier, while held-out success rises from 27.0% after round 1 to 65.8%. R-Zero

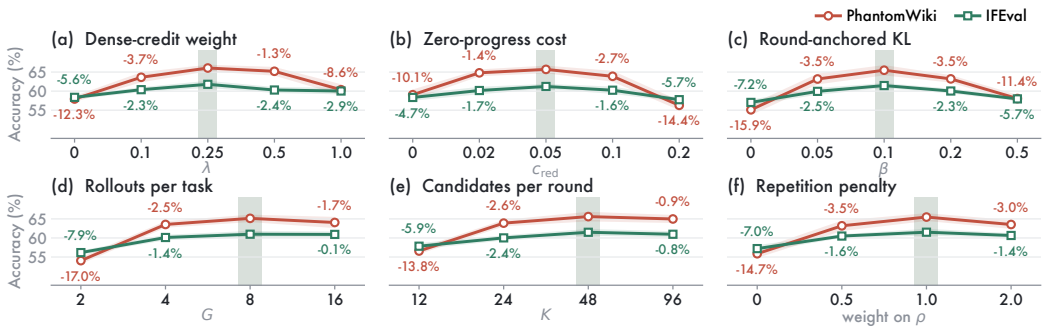


Figure 5: Hyperparameter sensitivity of the Qwen2.5-7B-Instruct unified model. Shaded columns indicate defaults, and annotations show relative drops from each curve's best.

keeps task difficulty nearly unchanged and reaches a lower final capture rate and held-out success. This behavior illustrates how LURE adapts absolute task difficulty to hold a fixed Solver-relative frontier as the Solver improves, rather than following a predefined difficulty schedule.

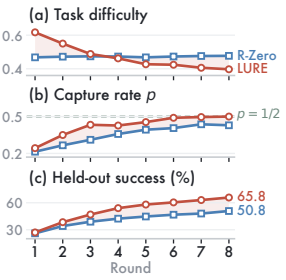


Figure 6: Anatomy of a chase on PhantomWiki.

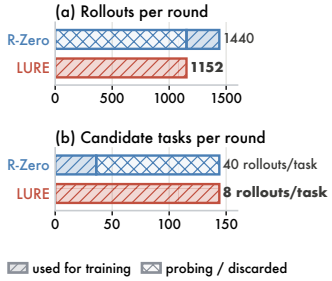


Figure 7: Per-round rollout and task budget.

Table 4: Share of rollout groups (%↓) whose G=8 outcomes are identical, counted over all candidate groups before filtering. Mean is over eight rounds.

Method	Round 1	2	4	8	Mean
GRPO	47.2	43.1	36.8	30.6	39.9
GRPO-Zero	47.9	43.1	54.2	56.2	48.3
R-Zero	47.2	34.7	34.7	26.4	35.6
LURE (Ours)	**45.1**	**33.3**	**24.3**	**20.1**	**26.0**
Ideal frontier	0.8	0.8	0.8	0.8	0.8

Curriculum Cost. Figure 7 compares the rollout costs of post-hoc filtering and adaptive task placement. Both methods emit 144 candidates per round. R-Zero spends 1152 probe rollouts to retain 36 tasks and discards the remaining 108 before training, whereas LURE eliminates probing and trains on all 144 tasks. Consequently, LURE trains on four times as many tasks at one-fifth the rollout cost per task, while reducing the total per-round cost from 1440 to 1152 rollouts.

Group-Relative Signal. Table 4 reports the fraction of rollout groups with identical terminal outcomes, which provide no outcome-only group-relative signal. Under LURE, this fraction decreases from 45.1% in round 1 to 20.1% in round 8, with a mean of 26.0% versus 35.6% for R-Zero. In contrast, GRPO-Zero becomes increasingly degenerate as training proceeds. Dense executor credit still separates turns within such outcome-degenerate groups.

5 CONCLUSION

This paper propose LURE, a verifier-grounded pursuit–evasion framework for zero-data self-play between a task-generating Challenger and a planner–executor Solver. A capture-frontier objective adapts task placement to verifier-confirmed Solver success, while capture-anchored dense process credit and round-anchored regularization support stable co-evolution. Across three verifiable reasoning environments and three backbone families, LURE consistently outperforms trained baselines under both unified and specialist settings. These gains extend to thirteen out-of-domain benchmarks, where LURE achieves the strongest overall transfer.

AI USE STATEMENT

We used large language model tools to polish the writing of this paper. We also used such tools to write the code for plotting statistical result figures and to check the consistency of mathematical notation. The authors take full responsibility for its content.

REPRODUCIBILITY STATEMENT

App. A gives every task generator with its difficulty mapping, verifier, role prompt, and baseline construction, together with the full optimizer configuration in Table 7 and the complete training round as Alg. 2. App. B specifies the in-domain evaluation sets, which are regenerated deterministically from fixed seeds, the metric definitions, the thirteen out-of-domain benchmarks with their decoding and answer extraction, and the gold-field redaction. App. C reports the full cross-environment scores and the difficulty-extrapolation protocol, and defines every ablation row. App. D states the assumptions of each theoretical result and gives complete proofs.

REFERENCES

Jasper Dekoninck, Nikola Jovanović, Tim Gehrunger, Kári Rögnvaldsson, Ivo Petrov, Chenhao Sun, and Martin Vechev. Beyond benchmarks: Matharena as an evaluation platform for mathematics with llms. *arXiv preprint arXiv:2605.00674*, 2026.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in neural information processing systems*, 33:13049–13061, 2020.

Guanting Dong, Keming Lu, Chengpeng Li, Tingyu Xia, Bowen Yu, Chang Zhou, and Jingren Zhou. Self-play with execution feedback: Improving instruction-following capabilities of large language models. In *International Conference on Learning Representations*, volume 2025, pp. 39286–39313, 2025.

Xeron Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, Minghao Liu, Yiming Liang, Xiaolong Jin, Zhenlin Wei, Chujie Zheng, et al. Supergpqa: Scaling llm evaluation across 285 graduate disciplines. *Advances in Neural Information Processing Systems*, 38, 2026.

Wenkai Fang, Shunyu Liu, Yang Zhou, Kongcheng Zhang, Tongya Zheng, Kaixuan Chen, Mingli Song, and Dacheng Tao. Serl: Self-play reinforcement learning for large language models with limited data. *Advances in Neural Information Processing Systems*, 38:103706–103738, 2026.

Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. *arXiv preprint arXiv:1705.06366*, 2017.

Albert Gong, Chao Wan, Kamilė Stankevičiūtė, Anmol Kabra, Raphael Thesmar, Johann Lee, Julius Klenke, Carla P Gomes, and Kilian Q Weinberger. Phantomwiki: On-demand datasets for reasoning and retrieval evaluation. In *ICML 2025 Workshop on Long-Context Foundation Models*, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-zero: Self-evolving reasoning llm from zero data. *arXiv preprint arXiv:2508.05004*, 2025a.

Ziyang Huang, Wangtao Sun, Jun Zhao, and Kang Liu. Improve rule retrieval and reasoning with self-induction and relevance reestimate. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 5473–5488, 2025b.

Mehran Kazemi, Bahare Fatemi, Hritik Bansal, John Palowitch, Chrysovalantis Anastasiou, Sanket Vaibhav Mehta, Lalit K Jain, Virginia Aglietti, Disha Jindal, Yuanzhu Peter Chen, et al. Big-bench extra hard. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 26473–26501, 2025.

Baohao Liao, Hanze Dong, Xinxing Xu, Christof Monz, and Jiang Bian. Self-hinting language models enhance reinforcement learning. *arXiv preprint arXiv:2602.03143*, 2026.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Bill Yuchen Lin, Ronan Le Bras, Kyle Richardson, Ashish Sabharwal, Radha Poovendran, Peter Clark, and Yejin Choi. Zebralogic: On the scaling limits of llms for logical reasoning. In *Forty-second International Conference on Machine Learning*, 2025.

Kaijing Ma, Xeron Du, Yunran Wang, Haoran Zhang, Zhoufutu Wen, Xingwei Qu, Jian Yang, Jiaheng Liu, Minghao Liu, Xiang Yue, et al. Kor-bench: Benchmarking language models on knowledge-orthogonal reasoning tasks. In *International Conference on Learning Representations*, volume 2025, pp. 80062–80161, 2025.

Jatin Prakash and Anirudh Buvanesh. What can you do when you have zero rewards during rl? *arXiv preprint arXiv:2510.03971*, 2025.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hanna Hajishirzi. Generalizing verifiable instruction following. *Advances in Neural Information Processing Systems*, 38, 2026.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL `https://arxiv.org/abs/2412.15115`.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv:2409.19256*, 2024.

Zhengxiang Shi, Qiang Zhang, and Aldo Lipani. Stepgame: A new benchmark for robust multi-hop spatial reasoning in texts. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 11321–11329, 2022.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L Hamilton. Clutrr: A diagnostic benchmark for inductive reasoning from text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4506–4515, 2019.

Zayne Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. Musr: Testing the limits of chain-of-thought with multistep soft reasoning. In *International Conference on Learning Representations*, volume 2024, pp. 14670–14728, 2024.

Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407*, 2017.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, 2024a.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, et al. Livebench: A challenging, contamination-limited llm benchmark. *arXiv preprint arXiv:2406.19314*, 2024.

Peng Xia, Kaide Zeng, Jiaqi Liu, Can Qin, Fang Wu, Yiyang Zhou, Caiming Xiong, and Huaxiu Yao. Agent0: Unleashing self-evolving agents from zero data via tool-integrated reasoning. *arXiv preprint arXiv:2511.16043*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022.

Qingyang Zhang, Haitao Wu, Changqing Zhang, Peilin Zhao, and Yatao Bian. Right question is already half the answer: Fully unsupervised llm reasoning incentivization. *Advances in neural information processing systems*, 38:67345–67372, 2026a.

Qingyang Zhang, Haitao Wu, Changqing Zhang, Peilin Zhao, and Yatao Bian. Right question is already half the answer: Fully unsupervised llm reasoning incentivization. *Advances in neural information processing systems*, 38:67345–67372, 2026b.

Andrew Zhao, Yiran Wu, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. *Advances in Neural Information Processing Systems*, 38:105816–105879, 2026a.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. Learning to reason without external rewards. In *International Conference on Learning Representations*, volume 2026, pp. 2548–2581, 2026b.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

Jiaming Zhou, Abbas Ghaddar, Ge Zhang, Liheng Ma, Yaochen Hu, Soumyasundar Pal, Mark Coates, Bin Wang, Yingxue Zhang, and Jianye Hao. Enhancing logical reasoning in large language models through graph-based synthetic data. *arXiv preprint arXiv:2409.12437*, 2024.

Yifei Zhou, Sergey Levine, Jason Weston, Xian Li, and Sainbayar Sukhbaatar. Self-challenging language model agents. *Advances in Neural Information Processing Systems*, 38:113959–113991, 2026.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Xinwei Long, Ermo Hua, Biqing Qi, et al. Ttrl: Test-time reinforcement learning. *Advances in Neural Information Processing Systems*, 38:131459–131483, 2026.

Roadmap. Appendix A presents the full experimental setup, including task generation, verification, role prompts, baselines, optimization settings, and the complete training procedure in Algorithm 2. Appendix B defines the evaluation datasets, metrics, and gold-field redaction. Appendix C reports cross-environment transfer, difficulty extrapolation, and per-environment ablations. Appendix D presents the theoretical results and full proofs.

A Experimental Setup

Task Generators and Difficulty Mapping. The Challenger emits only the scalar d, and the environment's programmatic generator instantiates the task from it. **Table 5** gives the mapping for each environment. All three clamp d to $[0, 1]$ and round it to a primary structural parameter with five levels, namely hop count, constraint count, or grid size, so a continuous d lets the Challenger express a target between levels while the repetition penalty acts on the realized structure.

Table 5: Difficulty mapping used by each task generator. $\lfloor \cdot \rceil$ denotes rounding to the nearest integer.

Environment	Structural parameter	d=0	d=1
PhantomWiki	hops $= 2 + \lfloor 4d \rceil$	2 hops	6 hops
	people $= 6 + \lfloor 14d \rceil$	6	20
	distractors $= \lfloor 10d \rceil$	0	10
IFEval	constraints $k = 1 + \lfloor 4d \rceil$	1	5
ZebraLogic	grid side $M = 2 + \lfloor 4d \rceil$	2×2	6×6

PhantomWiki (Gong et al., 2025) uses a vocabulary of 50 first names and 8 functional relations. Each instance first constructs a unique k-hop reasoning chain and then adds distractors that share the relation of a chain step but differ in subject, requiring entity-specific rather than relation-only retrieval. IFEval (Zhou et al., 2023) draws constraints from 16 programmatically verifiable types, instantiates them with the official constraint builder, and removes incompatible combinations according to the official conflict table. The requested k thus serves as an upper bound, with the realized constraint count recorded for each instance. ZebraLogic (Lin et al., 2025) generates logic-grid puzzles with unique solutions. The interaction horizons are set to $T_e = \text{hops} + 2$, $T_e = k + 2$, and $T_e = 4$ for PhantomWiki, IFEval, and ZebraLogic, respectively.

Verifiers and Progress Functions. Each environment provides a binary terminal capture signal $R(\tau) \in \{0, 1\}$ and a running progress score $\phi_t \in [0, 1]$, defined as the best verified progress achieved by turn t. By construction, ϕ_t is monotone, so $\Delta\phi_t \geq 0$, and regressions in the current attempt receive zero rather than negative progress. For a task decomposed into H verifiable units,

$$\phi_t = \max_{t' \leq t} \frac{|\{h \leq H : \text{unit } h \text{ is verified at turn } t'\}|}{H}, \tag{13}$$

with $\phi_t = 1$ upon capture. A verification unit corresponds to one resolved relation hop in PhantomWiki, one satisfied instruction constraint in IFEval, and one correct grid cell in ZebraLogic.

Difficulty Signatures. The repetition penalty $\rho(x)$ of Eq. 4 is computed over coarse per-environment signatures: the pair (hop count, population size) for PhantomWiki, the constraint count for IFEval, and the grid dimensions for ZebraLogic. Two emissions collide when their signatures are identical, regardless of surface content.

Challenger Prompts. The Challenger receives a system and user message per environment and emits a single difficulty tag. Two conditional segments are appended to every user message.

Challenger, PhantomWiki

System. You are a curriculum designer for a multi-hop relational QA agent (answer questions by composing facts about people). Pick the difficulty at the learner's frontier.

User. Choose the difficulty of the NEXT multi-hop question (harder = more reasoning hops, more people, more distractor facts). Output EXACTLY: `<task>difficulty: <number 0.0-1.0></task>`

Challenger, IFEval

System. You are a curriculum designer for an instruction-following agent (write a response satisfying verifiable constraints). Pick the difficulty at the learner's frontier.

User. Choose the difficulty of the NEXT instruction-following task (harder = more simultaneous verifiable constraints). Output EXACTLY: `<task>difficulty: <number 0.0-1.0></task>`

Challenger, ZebraLogic

System. You are a curriculum designer for a logic-grid puzzle agent (zebra puzzles: deduce the unique house-by-attribute grid from clues). Pick the difficulty at the learner's frontier.

User. Choose the difficulty of the NEXT logic-grid puzzle (harder = more houses and attributes). Output EXACTLY: `<task>difficulty: <number 0.0-1.0></task>`

The first appended segment is a cold-start band hint, present in every round and every slot. The hint spreads the slots across the difficulty axis from the first round, so the frontier reward has variance to learn from.

Appended to every Challenger prompt

(curriculum slot {`uid`}: aim NEAR difficulty {`band`} (you decide the exact value); slots must COVER a wide range.)

with band $= f + (0.95 - f) \cdot \frac{(3\,\mathrm{uid}) \bmod 7}{6}$ *and floor* $f = 0.15$.

ZebraLogic sets the floor to 0 so that the first round includes the 2×2 grid. The second segment is a competence summary produced from the previous round, appended from round two onward.

Appended from round two onward

(Solver success by difficulty (last round): d=0.0: 45% (12 eps) | d=0.2: 30% (18 eps) | . . . – favor difficulties where success is 30-70%, the learning frontier; keep some coverage elsewhere.)

The competence summary partitions task difficulty into bins of width 0.2 and reports the terminal capture rate within each bin. Each proposal is limited to 384 tokens.

Validity Gate. A generated task enters the curriculum only after passing an environment-specific validity check. PhantomWiki recomputes the reasoning chain from the instantiated facts and requires each hop to resolve functionally to the gold answer, with at least two hops. IFEval requires non-empty constraint specifications and prompt content. ZebraLogic checks consistency between the declared grid dimensions and the instantiated rows and columns, with all cells populated. Proposals without a valid difficulty value or failing this check receive the Challenger reward $-1 - \rho(x)$.

Planner and Executor Prompts. The planner template is shared across all three environments and is conditioned on the environment's predicate library.

Planner, all environments

You are the PLANNER in a verifier-grounded planning system. Decompose the task into an ORDERED list of VERIFIABLE subgoals. Each subgoal MUST be tied to exactly one `predicate_id` from the environment's predicate library: a subgoal counts as achieved ONLY when its predicate is verified satisfied. Do not invent rewards.

1. Output the plan and NOTHING else between `<plan>` and `</plan>`.
2. One subgoal per line, ordered, as `<n>. [predicate_id] <subgoal text>`, copying the predicate verbatim.
3. Keep subgoals minimal, ordered by dependency, and each independently verifiable.
4. On a replan, revise ONLY the parts whose predicates are unsatisfied or regressed.

The user message is assembled in a fixed order: task specification, machine-checkable constraints, predicate library, and on a replan the current state, the previous plan, and the verifier feedback. The executor has one default template plus two environment-specific variants, because both ZebraLogic and IFEval score a single action against the whole target rather than accumulating credit across actions.

Executor, default

You are the EXECUTOR in a verifier-grounded planning system. Given a PLAN and the CURRENT observation, take EXACTLY ONE action toward the first not-yet-satisfied subgoal.

1. Output exactly one action wrapped in `<action>...</action>` and nothing else.
2. The action MUST be one of the available actions this turn.
3. Take only ONE action. Do not chain multiple actions.
4. If a constraint regressed, prefer the action that recovers the violated predicate.

Executor, environment variants

ZebraLogic. The action must contain the ENTIRE grid, one line per house in order, as `House i: <value> | <value> | ...`. A partial grid scores ZERO. Do not guess a value that a clue determines.

IFEval. The action must contain the COMPLETE response, that is the actual prose rather than a command. Every constraint is checked against that text, and a partial or placeholder response scores ZERO.

Baselines. The four baselines isolate the three ingredients that LURE combines: training on verifiable tasks, generating those tasks without human data, and positioning them at the Solver's frontier. **Table 6** contrasts them along these axes under a shared backbone, rollout group, round budget, and evaluation protocol.

Table 6: Baselines compared with LURE.

	Task supply		Solver training and cost	
Method	**Training tasks**	**Curriculum signal**	**Solver reward**	**Selection cost**
Base	none	none	none	none
GRPO	1,152 curated tasks	fixed pool	verifier capture	none
GRPO-Zero	self-generated pool from round 1	frozen after round 1	as R-Zero	round 1 only
R-Zero	Challenger-authored problem text	uncertainty reward and band filter	majority-vote pseudo-label	G probe rollouts per candidate
LURE (Ours)	generator instances at a selected difficulty d	capture-frontier reward	capture and verified progress	none

- **Base.** The released instruction-tuned checkpoint without additional training, serving as the zero-shot reference for each backbone.
- **GRPO.** Group-relative policy optimization (Shao et al., 2024) trained on 1,152 externally curated tasks, sampled equally from the public training splits of the three environments and disjoint from all evaluation sets. This matches the 144 tasks per round over 8 rounds used by LURE.
- **GRPO-Zero.** A fixed-curriculum control. The task pool is generated in round 1 following the R-Zero procedure and reused in all subsequent rounds, so the training distribution stays fixed as the Solver evolves.
- **R-Zero.** We implement the proposer–solver co-evolution procedure of R-Zero (Huang et al., 2025a), with both roles optimized by GRPO. The Challenger receives

$$\max\big(0,\, 1 - 2\,\big|\hat{p}(x) - \tfrac{1}{2}\big| - r_{\mathrm{rep}}(x)\big)\,,$$

where $\hat{p}(x)$ denotes the fraction of Solver samples agreeing with the majority-vote answer and $r_{\mathrm{rep}}(x)$ penalizes near-duplicate proposals. A task is retained for Solver training when $\big|\hat{p}(x) - \frac{1}{2}\big| \le 0.25$, and Solver rewards are defined by agreement with the majority-vote pseudo-label. We preserve these selection and reward rules while matching the candidate

and rollout budgets of LURE, using 144 candidates per round and $G = 8$ probe rollouts per candidate.

Optimizer Configurations. **Table 7** lists the settings. Baselines share the settings in the Baselines column, following R-Zero where applicable, and LURE runs its stabilized recipe, whose contribution the *w/o* stabilized recipe row of **Table 11** isolates by running LURE under the baseline settings.

Table 7: Optimizer settings per method.

	Base	**Baselines**	**LURE (Ours)**
Learning rate	/	10^{-5}	5×10^{-6}
PPO epochs	/	2	1
KL coefficient β	/	0	0.1
KL anchor	/	none	round-start
Rollout group G	/	8	8
Clip range ϵ	/	0.2	0.2
Candidates per environment K	/	48	48
Dense-credit weight λ	/	/	0.25
Zero-progress cost c_{red}	/	/	0.05
Repetition-penalty weight on ρ	/	/	1
Challenger rollout temperature	/	1.0	1.0
Solver rollout temperature	/	0.7	0.7
Rollout top-p	/	0.95	0.95
Max executor turns $T_{\max}$ / replans	/	20 / 3	20 / 3
Warm-start seeding	/	off	off
Self-play rounds	/	8	8

Full Round. **Alg. 2** expands **Alg. 1** into the complete training round.

B EVALUATION PROTOCOL

In-Domain Evaluation Sets. **Table 8** lists the three in-domain evaluation sets. PhantomWiki instances are regenerated from fixed seeds with one near-miss distractor per hop, and instances that fail the well-posedness check are discarded. ZebraLogic puzzles are drawn from the labeled grid-mode set with a fixed seed. Training tasks use disjoint generator seed ranges, and all sets are reproduced deterministically, so every method sees identical instances. Evaluation uses greedy decoding with a single rollout per instance, and only PhantomWiki uses multi-turn interaction.

Metric Definitions. Let $\mathcal{D}$ be an evaluation set. PhantomWiki reports chain-verified exact match,

$$\text{SUCCESS} = \frac{1}{|\mathcal{D}|} \sum_{q \in \mathcal{D}} \mathbb{1}[\phi_T(q) = 1], \tag{14}$$

IFEval reports strict per-prompt accuracy, scoring an instance as correct only when all of its constraints hold jointly. ZebraLogic reports both puzzle accuracy, which applies the same all-or-nothing rule to the full grid, and cell accuracy, the mean fraction of correctly filled cells,

$$\text{Cell} = \frac{1}{|\mathcal{D}|} \sum_{q \in \mathcal{D}} \phi_T(q). \tag{15}$$

General reasoning in **Table 1** follows the chain-of-thought evaluation of R-Zero (Huang et al., 2025a).

Out-of-Domain Benchmarks. Table 9 summarizes the thirteen out-of-domain benchmarks and their evaluation protocols. For mathematics benchmarks, we sample 16 responses per problem with temperature 0.6, top-p 0.95, and a maximum generation length of 8,192 tokens, and report Avg@16

Algorithm 2 One round of LURE

Require: Challenger π_C, planner π_{plan}, executor π_{exec}, environments $\mathcal{E}$, prompt slots U, samples per slot V, rollouts G, weights λ, c_{red}, β, clip ϵ, difficulty floor f
Ensure: updated π_C, π_{plan}, π_{exec}

1. Freeze the round

$\bar{\pi}_r \leftarrow \pi_r$ for $r \in \{C, \text{plan}, \text{exec}\}$, one snapshot serving as behaviour policy, KL reference and proximal centre
task pool $\mathcal{X} \leftarrow \emptyset$, buffers $\mathcal{D}_C, \mathcal{D}_{\text{plan}}, \mathcal{D}_{\text{exec}} \leftarrow \emptyset$

2. The Challenger places the curriculum

for each environment $e \in \mathcal{E}$ **do**
  for each prompt slot $u = 1, \ldots, U$ **do**
    band $\leftarrow f + (0.95 - f) \cdot \big((3u) \bmod 7\big)/6$
    draw V difficulty tags $d \sim \bar{\pi}_C(\cdot \mid e, \text{band}, \text{competence summary of the previous round})$
    for each tag **do**
      read the last `<task>` segment and clamp d to $[0, 1]$, marking the emission invalid if it fails to parse
      instantiate $x \leftarrow g_e(d, \omega)$ and add it to $\mathcal{X}$ only if the per-environment structural check passes
    end for
  end for
end for

3. The Solver attempts every task

for each task $x \in \mathcal{X}$ **do**
  for $g = 1, \ldots, G$ **do**
    plan with $\bar{\pi}_{\text{plan}}$, then run $\bar{\pi}_{\text{exec}}$ for at most T_e turns, replanning when a predicate regresses
    the verifier returns capture $R(\tau_g)$ and the progress trace $\{\phi_t\}$, monotone so $\Delta\phi_t \geq 0$
  end for
  $p(x) \leftarrow \frac{1}{G} \sum_g \mathbb{1}[R(\tau_g) = 1]$
end for

4. One verifier signal becomes three training signals

$\rho(x) \leftarrow (n_{\text{sig}}(x) - 1)/N$, counting the N emissions of environment e in the round that share x's difficulty signature
$r_C(x) \leftarrow \min(p(x), 1 - p(x)) - \rho(x)$ for well-formed x and $-1 - \rho(x)$ otherwise, then $A^C \leftarrow z_{\text{slot}}(r_C)$
$r_t \leftarrow R(\tau)\mathbb{1}[t = T_\tau] + \lambda\big(\Delta\phi_t - c_{\text{red}}\mathbb{1}[\Delta\phi_t = 0]\big)$, then $A_t^{\text{exec}} \leftarrow z_{\mathcal{G}(x)}(r_t)$
$r^{\text{plan}}(\tau) \leftarrow \sum_i \Phi_i$ over the executor segments that plan governed, then $A_\tau^{\text{plan}} \leftarrow z_{\mathcal{T}(x)}(r^{\text{plan}}(\tau))$
append the Challenger sample to $\mathcal{D}_C$, the executor turns to $\mathcal{D}_{\text{exec}}$, the planner rollouts to $\mathcal{D}_{\text{plan}}$

5. One proximal step per role

for $r \in \{C, \text{plan}, \text{exec}\}$ **do**
  $\varrho_r \leftarrow \pi_{\theta_r}(y \mid s)/\bar{\pi}_r(y \mid s)$, the token-level ratio to this round's snapshot
  take a single epoch on $\mathcal{D}_r$ maximizing $\mathcal{J}_r(\theta_r)$ in Eq. 11
end for
return π_C, π_{plan}, π_{exec}

as the mean correctness across samples. Final answers are extracted from the last boxed expression and evaluated for mathematical equivalence using a fixed extractor and scorer shared by all methods and aligned with the cited Base evaluations. MuSR uses greedy decoding. LiveBench follows the 2024-11-25 release and official scorers, KOR-Bench uses its category-specific templates and official answer extractor, and BBEH reports micro accuracy.

Gold-Field Redaction. The planner prompt omits the four fields below.

Table 8: In-domain evaluation sets, reported in **Table 1**.

Environment	n	Metric	Instance construction
PhantomWiki	120	chain-verified exact match	generated at difficulties 0.20 to 0.80, 3 to 5 hops, 9 to 17 people
IFEval	541	strict per-prompt accuracy	official prompt file, single turn
ZebraLogic	300	puzzle accuracy and cell accuracy	stratified over the 25 grid sizes from 2×2 to 6×6

Table 9: Out-of-domain benchmarks used for the transfer evaluation, grouped as in **Table 3**.

Benchmark	n	Metric and protocol	Base score
Near-Domain			
CLUTRR (Sinha et al., 2019)	1,048	answer match, direct answer	Huang et al. (2025b)
LiveBench-IF (White et al., 2024)	200	official composite score	White et al. (2024)
IFBench (Pyatkin et al., 2026)	300	prompt-level loose	Pyatkin et al. (2026)
LiveBench-Reasoning (White et al., 2024)	150	official score	White et al. (2024)
Far Transfer: Mathematics			
MATH500 (Hendrycks et al., 2021; Lightman et al., 2024)	500	Avg@16, boxed answer	Liao et al. (2026)
AIME24 (Dekoninck et al., 2026)	30	Avg@16, boxed answer	Liao et al. (2026)
AIME25 (Dekoninck et al., 2026)	30	Avg@16, boxed answer	Liao et al. (2026)
AMC23	40	Avg@16, boxed answer	Liao et al. (2026)
OlympiadBench (He et al., 2024)	675	Avg@16, English open-ended subset	Liao et al. (2026)
Far Transfer: Logic			
KOR-Bench (Ma et al., 2025)	1,250	accuracy, zero-shot with rules	Ma et al. (2025)
BBEH (Kazemi et al., 2025)	4,520	micro accuracy	Kazemi et al. (2025)
Far Transfer: Other Reasoning			
StepGame (Shi et al., 2022)	900	accuracy, five-shot	Zhou et al. (2024)
MuSR (Sprague et al., 2024)	756	accuracy, official prompts	Sprague et al. (2024)

Redacted fields and what they held

`answer` (PhantomWiki, the gold entity), `table_rows` (ZebraLogic, the complete solution grid), `gold` and `solution` (generic).

Verification reads the full task record.

C Additional Analysis

Cross-Environment Transfer. **Table 10** reports the scores behind **Fig. 3**, including ZebraLogic cell accuracy, which the figure omits. Specialists are evaluated with the in-domain protocol of **App. B**.

Table 10: Cross-environment transfer (%) of per-environment specialists on Qwen2.5-7B-Instruct. Columns name the environment a specialist was trained on and rows name the evaluation. Best and second-best per training environment: boxed and underlined.

	Base	GRPO			R-Zero			LURE (Ours)		
Evaluated on		PhantomWiki	IFEval	ZebraLogic	PhantomWiki	IFEval	ZebraLogic	PhantomWiki	IFEval	ZebraLogic
PhantomWiki (Succ.)	25.0	60.8	28.3	27.5	56.7	27.5	25.0	**80.0**	**31.7**	**39.2**
IFEval (Prompt)	63.0	**59.7**	62.8	59.0	56.0	60.1	60.3	58.4	**63.0**	**63.8**
ZebraLogic (Puzzle)	11.6	**15.1**	12.0	16.1	11.3	11.3	15.0	11.4	**12.7**	**18.3**
ZebraLogic (Cell)	42.3	41.7	43.1	46.5	43.9	42.3	44.8	**45.0**	**43.4**	**48.6**

Difficulty Extrapolation. Table 2 evaluates 100 fixed valid instances per environment at each $d \in \{1, 1.25, 1.5\}$, shared across methods and disjoint from training. The $d = 1$ rows correspond to the hardest training level and therefore differ from the mixed-difficulty results in Table 1. For extrapolation, the specified difficulty is passed directly to the generator without the training-time clamp to $[0, 1]$. Generation preserves the validity criteria of each environment, including realized

constraint counts for IFEval, unique solutions for ZebraLogic, and valid functional chains for PhantomWiki. As in training, each instance is evaluated with one planner–executor episode under the same task-specific horizon for all methods: $8/9/10$ turns for PhantomWiki, $7/8/9$ for IFEval, and 4 for ZebraLogic, subject to the global limits of 20 steps and 3 replans. PhantomWiki and IFEval report episode success over 100 instances, while ZebraLogic reports the mean fraction of verifier-confirmed correct attribute cells.

Table 11: Per-environment ablation results (%) on the Qwen2.5-7B-Instruct unified model. Δ is the change in mean against the full system. Best and second-best results: boxed and underlined.

Configuration	PhantomWiki	IFEval	ZebraLogic (Puzzle)	Mean	Δ
Full	**65.8**	**61.2**	**16.9**	**48.0**	/
Curriculum					
w/o learned challenger	40.0	58.0	14.1	37.4	−10.6
w/o capture-frontier reward	53.3	58.0	15.4	42.2	−5.8
w/o repetition penalty	55.8	56.9	14.6	42.4	−5.6
Solver credit					
w/o dense credit	57.5	57.9	15.1	43.5	−4.5
w/o zero-progress cost	59.2	58.2	15.8	44.4	−3.6
w/o planner credit	61.7	59.5	16.2	45.8	−2.2
Stabilization					
w/o round-anchored KL	55.0	56.7	13.4	41.7	−6.3
w/o stabilized recipe	52.5	55.5	12.9	40.3	−7.7

Ablation Configurations. Each row of **Table 11** modifies one component while holding the remaining settings fixed, as follows.

- *w/o* learned challenger: the Challenger is replaced by programmatic sampling with the $[0.25, 0.75]$ acceptance band, keeping the dense credit and the stabilized recipe.
- *w/o* capture-frontier reward: the Challenger is trained on a reward that increases monotonically with difficulty instead of peaking at $p = 1/2$.
- *w/o* repetition penalty: $\rho(x)$ is set to zero.
- *w/o* dense credit: $\lambda = 0$, leaving the terminal capture alone.
- *w/o* zero-progress cost: $c_{\text{red}} = 0$.
- *w/o* planner credit: the planner receives the executor's credit rather than Eq. 9.
- *w/o* round-anchored KL: $\beta = 0$.
- *w/o* stabilized recipe: learning rate 10^{-5}, two PPO epochs, and $\beta = 0$.

D Theorems and Proofs

D.1 Preliminaries: Group-Relative Policy Optimization

GRPO (Shao et al., 2024) replaces the value network of PPO (Schulman et al., 2017) with a baseline computed inside each group of responses. For one prompt the behaviour policy draws a group of G responses, the environment scores them, and each score is standardized against its group, so response g carries the advantage $A_g = z(r)_g$, where z sends a tuple v of length $n \geq 2$ to $z(v)_i = (v_i - \bar{v})/s(v)$, and to 0 when $s(v) = 0$, under either variance convention. Every token of a response inherits its advantage, and the policy maximizes the clipped surrogate

$$\begin{aligned} \mathcal{J}(\theta) = \mathbb{E}\big[\min\big(\varrho A,\ \mathrm{clip}(\varrho, 1{-}\epsilon, 1{+}\epsilon)\, A\big)\big] \\ - \beta\, D_{\mathrm{KL}}\big(\pi_\theta \,\|\, \pi_{\mathrm{ref}}\big), \end{aligned} \tag{16}$$

with ϱ the token-level probability ratio between π_θ and the behaviour policy, ϵ the clip range, and π_{ref} the reference of the KL penalty.

Throughout the proofs the three policies are their round-start snapshots and every quantity is conditioned on them. We write $q = \Pr(R(\tau) = 1 \mid x)$ for a task's capture probability under the Solver

snapshot, so the empirical capture rate $p(x)$ of Eq. 1 is an average of G independent $\mathrm{Bernoulli}(q)$ outcomes. The verifier emits $R(\tau) \in \{0,1\}$ and a best-so-far $\phi_t \in [0,1]$, which is nondecreasing, with $\phi_0 = 0$ and $\phi_{T_\tau} = 1$ when τ captures. Every trajectory has at most $T_{\max}$ executor turns, the global turn cap.

D.2 The Capture Frontier Maximizes the Group-Relative Signal

Proposition 1 (Frontier maximizes the group-relative signal, restated)**.** *Fix a task x with capture probability $q \in [0,1]$, and let $R_1, \ldots, R_G$ be its $G \geq 2$ rollout outcomes, independent and $\mathrm{Bernoulli}(q)$. Let $A_g = z(R)_g$ be the outcome-only advantages under the convention $s^2 = (G-1)^{-1}\sum_g (R_g - \bar{R})^2$. Then*

$$\sum_{g=1}^{G} A_g^2 = (G-1)\,\mathbb{1}[s(R) > 0] \quad \textit{pointwise}, \tag{17}$$

so that, with $\psi_G(q) = 1 - q^G - (1-q)^G$,

$$\mathbb{E}\left[\sum_{g=1}^{G} A_g^2\right] = (G-1)\,\psi_G(q). \tag{18}$$

Every A_g is nonzero if and only if the outcomes are not all equal. Both ψ_G and the frontier term $f(q) = \min(q, 1-q)$ are uniquely maximized on $[0,1]$ at $q = \frac{1}{2}$, where $\psi_G(\frac{1}{2}) = 1 - 2^{1-G}$. Hence the frontier term peaks at the capture probability at which the Solver's expected outcome-only signal is largest.

Proof. Since $s(R)^2 = (G-1)^{-1}\sum_g (R_g - \bar{R})^2$ is a sum of squares divided by $G-1 > 0$, we have $s(R) = 0$ if and only if $R_g = \bar{R}$ for every g, that is, if and only if $R_1 = \cdots = R_G$. Write D for this event.

D is the disjoint union of $\{R_g = 1\ \forall g\}$ and $\{R_g = 0\ \forall g\}$, disjoint because $0 \neq 1$. By independence, $\Pr(D) = q^G + (1-q)^G$ and therefore $\Pr(D^c) = \psi_G(q)$.

On D every $A_g = 0$ by the normalization convention, so the left side of Eq. 17 is 0. On D^c we have $s(R) > 0$ by the equivalence just shown, and $\bar{R} \in (0,1)$ while $R_g \in \{0,1\}$, so every $A_g \neq 0$. Moreover

$$\sum_{g=1}^{G} A_g^2 = \frac{1}{s(R)^2}\sum_{g=1}^{G}(R_g - \bar{R})^2 = \frac{(G-1)\,s(R)^2}{s(R)^2} = G - 1, \tag{19}$$

the middle equality being the definition of $s(R)^2$ rearranged. This proves Eq. 17, and taking expectations against $\Pr(D^c) = \psi_G(q)$ gives Eq. 18. Under the convention $s^2 = G^{-1}\sum_g (\cdot)^2$ the constant $G-1$ becomes G and nothing else changes.

ψ_G is a polynomial with $\psi_G'(q) = G[(1-q)^{G-1} - q^{G-1}]$. Since $G \geq 2$, the exponent $G-1 \geq 1$ and $u \mapsto u^{G-1}$ is strictly increasing on $[0,1]$, so for $q \in [0,1]$,

$$\begin{aligned} \psi_G'(q) > 0 \iff (1-q)^{G-1} > q^{G-1} \\ \iff 1 - q > q \iff q < \tfrac{1}{2}, \end{aligned} \tag{20}$$

and symmetrically $\psi_G'(q) < 0$ for $q > \frac{1}{2}$, with $\psi_G'(\frac{1}{2}) = 0$. Hence ψ_G strictly increases on $[0, \frac{1}{2}]$ and strictly decreases on $[\frac{1}{2}, 1]$, so $\frac{1}{2}$ is its unique maximizer, with value $1 - 2\cdot 2^{-G} = 1 - 2^{1-G}$.

Using $\min(a,b) = \frac{1}{2}(a+b-|a-b|)$, $f(q) = \frac{1}{2}(1-|2q-1|)$, and $|2q-1| \geq 0$ with equality exactly at $q = \frac{1}{2}$. So f too is uniquely maximized at $\frac{1}{2}$, with value $\frac{1}{2}$. The two functions therefore share the unique maximizer $\frac{1}{2}$, and Eq. 18 is a positive constant times ψ_G, which completes the proof. □

Corollary 1. *At $G = 8$, the probability that all outcome-only advantages of a group vanish is $2^{-7} = 0.0078$ at $q = \frac{1}{2}$ and $0.1^8 + 0.9^8 = 0.4305$ at $q = 0.1$, a factor of 55.1. The expected signal $\mathbb{E}[\sum_g A_g^2]$ falls correspondingly from 6.945 to 3.987.*

Lemma 1. *Let $G \geq 2$, let $Y \sim \mathrm{Binomial}(G, q)$, and set $h(q) = \mathbb{E}[\min(Y/G,\, 1 - Y/G)]$, the expected frontier term of the empirical capture rate. Then h is uniquely maximized on $[0, 1]$ at $q = \frac{1}{2}$.*

Proof. Write $g(u) = \min(u, 1-u)$, concave on $[0,1]$ as a minimum of two affine functions, and symmetric in the sense $g(1-u) = g(u)$. Then $h = B_G g$, where $B_G g(p) = \sum_{k=0}^{G} g(k/G)\binom{G}{k}p^k(1-p)^{G-k}$ is the Bernstein operator.

Concavity. We derive shape preservation directly. Differentiating a single Bernstein basis element and using $k\binom{n}{k} = n\binom{n-1}{k-1}$ and $(n-k)\binom{n}{k} = n\binom{n-1}{k}$,

$$\begin{aligned}\tfrac{d}{dp}\Big[\tbinom{n}{k}p^k(1-p)^{n-k}\Big] = n\Big[&\tbinom{n-1}{k-1}p^{k-1}(1-p)^{n-k} \\ &-\tbinom{n-1}{k}p^k(1-p)^{n-k-1}\Big].\end{aligned} \tag{21}$$

Here and below, binomial coefficients with an index outside their range are taken as zero. Summing against $f(k/n)$ and reindexing the first sum by $j = k-1$, the two sums combine into

$$(B_n f)'(p) = n\sum_{j=0}^{n-1}\Big[f\big(\tfrac{j+1}{n}\big) - f\big(\tfrac{j}{n}\big)\Big]\tbinom{n-1}{j}p^j(1-p)^{n-1-j}. \tag{22}$$

Applying Eq. 22 twice gives

$$(B_n f)''(p) = n(n-1)\sum_{k=0}^{n-2}\big(\Delta_k^2 f\big)\tbinom{n-2}{k}p^k(1-p)^{n-2-k}, \tag{23}$$

with $\Delta_k^2 f = f\big(\frac{k+2}{n}\big) - 2f\big(\frac{k+1}{n}\big) + f\big(\frac{k}{n}\big)$. Concavity of g applied to the midpoint of $\frac{k}{n}$ and $\frac{k+2}{n}$ gives $\Delta_k^2 g \leq 0$ for every k, and the basis elements are nonnegative on $[0,1]$, so $(B_G g)'' \leq 0$ and h is concave.

Symmetry. Substituting $j = G - k$ and using $\binom{G}{G-j} = \binom{G}{j}$ together with $g(1-u) = g(u)$,

$$\begin{aligned}h(1-p) &= \sum_k g(k/G)\tbinom{G}{k}(1-p)^k p^{G-k} \\ &= \sum_j g(j/G)\tbinom{G}{j}p^j(1-p)^{G-j} = h(p).\end{aligned} \tag{24}$$

Maximizer. By concavity and symmetry, for every p, $h(\frac{1}{2}) = h\big(\frac{p+(1-p)}{2}\big) \geq \frac{1}{2}h(p) + \frac{1}{2}h(1-p) = h(p)$.

Uniqueness. Suppose $h(p_0) = h(\frac{1}{2})$ for some $p_0 \neq \frac{1}{2}$. A concave function attaining its maximum at two points is constant on the segment joining them, so h is constant on a nondegenerate interval. Being a polynomial, it is then constant on all of $[0,1]$. But $h(0) = g(0) = 0$, whereas at $p = \frac{1}{2}$ every term of h is nonnegative and the term $k = 1$ alone contributes $g(\frac{1}{G})\binom{G}{1}2^{-G} = \frac{1}{G}\cdot G \cdot 2^{-G} = 2^{-G}$, so $h(\frac{1}{2}) \geq 2^{-G} > 0$. This contradiction forces $p_0 = \frac{1}{2}$. □

Eq. 17 holds because capture is binary, which is exactly the regime of terminal verification, and it is what reduces the frontier to a statement about a single scalar: once a group is non-degenerate its squared advantage mass is fixed at $G - 1$, so placement controls the signal entirely through the degeneracy probability.

D.3 Telescoping, Capture Dominance, and the Length Bias

Proposition 2 (Capture-preserving dense credit)**.** *Let τ have executor turns $t = 1, \ldots, T_\tau$ with $T_\tau \leq T_{\max}$, let $Z(\tau) = |\{t \leq T_\tau : \Delta\phi_t = 0\}|$, write $c = c_{\mathrm{red}}$, and assume $\lambda > 0$ and $c \geq 0$. Since ϕ is best-so-far, the trajectory total of Eq. 6 is*

$$U(\tau) \;=\; \sum_{t=1}^{T_\tau} r_t \;=\; R(\tau) + \lambda\,\phi_{T_\tau} - \lambda c\,Z(\tau). \tag{25}$$

Consequently: (i) *at $c = 0$ the dense credit contributes $\lambda\phi_{T_\tau}$, a function of terminal progress alone.* (ii) *If $\lambda c\,(T_{\max} - 1) < 1$ then $U(\tau) > U(\tau')$ for every capturing τ and every non-capturing τ'.* (iii) *If $\tilde{\tau}$ is obtained from τ by inserting one zero-progress turn while preserving R and ϕ_{T_τ}, then $U(\tilde{\tau}) - U(\tau) = -\lambda c \le 0$, whereas under a variant that credits R on every turn the difference is $R(\tau) - \lambda c$, strictly positive for a capturing trajectory whenever $\lambda c < 1$.*

Proof. The potential ϕ is nondecreasing with $\phi_0 = 0$, so $\Delta\phi_t \ge 0$ and $\sum_{t=1}^{T_\tau} \Delta\phi_t = \phi_{T_\tau} - \phi_0 = \phi_{T_\tau}$ by telescoping. Summing Eq. 6 over t, the terminal term contributes $R(\tau)$ exactly once because $t = T_\tau$ occurs once, giving Eq. 25. Claim (i) is immediate at $c = 0$, the increment $\lambda\Delta\phi_t$ being the potential difference of ϕ across one turn, so the dense credit accumulates to a function of terminal progress alone.

Let τ capture, so $R(\tau) = 1$ and $\phi_{T_\tau} = 1$. By telescoping $\sum_{t=1}^{T_\tau} \Delta\phi_t = 1 > 0$, so some turn has $\Delta\phi_t > 0$, and hence $Z(\tau) \le T_\tau - 1 \le T_{\max} - 1$.

For capturing τ, Eq. 25 with $Z(\tau) \le T_{\max} - 1$ gives $U(\tau) \ge 1 + \lambda - \lambda c\,(T_{\max} - 1)$. For non-capturing τ' we have $R(\tau') = 0$, $\phi_{T_{\tau'}} \le 1$ and $Z(\tau') \ge 0$, so $U(\tau') \le \lambda$. Subtracting,

$$U(\tau) - U(\tau') \;\ge\; 1 - \lambda c\,(T_{\max} - 1) \;>\; 0 \tag{26}$$

by hypothesis, and τ, τ' were arbitrary.

For (iii), $\tilde{\tau}$ has the same R and terminal progress as τ and one more zero-progress turn, so $Z(\tilde{\tau}) = Z(\tau) + 1$ and U changes by $-\lambda c$. Under the replicated-terminal variant the total is $T_\tau R(\tau) + \lambda\phi_{T_\tau} - \lambda c Z(\tau)$, so the same append changes it by $R(\tau) - \lambda c$, which is $1 - \lambda c > 0$ when τ captures and $\lambda c < 1$. □

Corollary 2. *With $\lambda = 0.25$, $c_{\mathrm{red}} = 0.05$ and $T_{\max} = 20$, we have $\lambda c_{\mathrm{red}}(T_{\max} - 1) = 0.2375 < 1$, so Proposition 2 (ii) applies with $4.21\times$ headroom. If a trajectory that verifies all H units of its task always captures, then $\phi_{T_{\tau'}} \le (H - 1)/H$ for a non-capturing τ', which relaxes the condition to $\lambda c\,(T_{\max} - 1) < 1 + \lambda/H$.*

Claim (iii) is what makes the single terminal credit necessary rather than cosmetic: replicating capture across turns would make the return grow with trajectory length, so a capturing policy could improve its score by padding with admissible zero-progress turns. Eq. 6 removes that incentive outright, and claim (ii) guarantees the dense term can never invert the capture ordering it is meant to densify.

D.4 Round Anchoring Makes the First Step Exact

Matching the token-level ratio of Eq. 16, we take (s, y) to range over the token-level pairs of the round's batch, so s is a context prefix and y the next token drawn from the finite vocabulary. Below, $\hat{\mathbb{E}}$ denotes the average over these batch pairs, $\bar{\pi}$ denotes the snapshot probabilities that enter the ratio, and the KL term sums over the full vocabulary at each batch context, so every quantity is a finite sum. We assume $\pi_\theta(y \mid s) > 0$ and $\pi_{\mathrm{ref}}(y \mid s) > 0$ for every batch context s and every vocabulary token y, with π_{ref} fixed independently of θ, and $\theta \mapsto \pi_\theta(y \mid s)$ twice continuously differentiable, as holds for a full-support softmax head. The prompt distribution is the fixed empirical distribution of the batch.

Proposition 3 (First-step consistency of round anchoring)**.** *Let $\bar{\theta}$ be a role's round-start parameters and $\bar{\pi} = \pi_{\bar{\theta}}$ the snapshot whose probabilities enter the ratio. Then, at $\theta = \bar{\theta}$:* (i) *$\varrho \equiv 1$, the clip is inactive on a neighbourhood, and the gradient of the surrogate is the empirical group-relative policy gradient $\hat{\mathbb{E}}[A\,\nabla_\theta \log \pi_\theta]$.* (ii) *If $\pi_{\mathrm{ref}} = \bar{\pi}$ then $\nabla_\theta D_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_{\mathrm{ref}}) = 0$, so the penalty does not tilt the first-order direction.* (iii) *If $\pi_{\mathrm{ref}} \ne \bar{\pi}$ then*

$$\nabla_\theta D_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_{\mathrm{ref}})\big|_{\bar{\theta}} = \hat{\mathbb{E}}_s\, \mathbb{E}_{y \sim \bar{\pi}(\cdot|s)} \left[\nabla_\theta \log \pi_\theta \cdot \log \frac{\bar{\pi}}{\pi_{\mathrm{ref}}} \right], \tag{27}$$

which carries no factor of A and is in general nonzero. (iv) *Finally, $\nabla^2_\theta D_{\mathrm{KL}}(\pi_\theta \,\|\, \bar{\pi})|_{\bar{\theta}}$ is the Fisher information $F(\bar{\theta})$, so β imposes a local Fisher-quadratic penalty centred on the behaviour policy.*

Proof. (i) At $\bar{\theta}$ the ratio is $\varrho = \pi_{\bar{\theta}}/\bar{\pi} = 1$ pointwise. Since $\epsilon > 0$ and $\theta \mapsto \varrho$ is continuous, there is a neighbourhood on which $\varrho \in (1-\epsilon, 1+\epsilon)$. There $\mathrm{clip}(\varrho, 1-\epsilon, 1+\epsilon) = \varrho$, both arguments of the minimum coincide, and the surrogate equals $\hat{\mathbb{E}}[\varrho A]$. Differentiating the finite sum, $\nabla_\theta \hat{\mathbb{E}}[\varrho A]|_{\bar{\theta}} = \hat{\mathbb{E}}[A\, \nabla_\theta \pi_\theta / \bar{\pi}]|_{\bar{\theta}} = \hat{\mathbb{E}}[A\, \nabla_\theta \log \pi_\theta]|_{\bar{\theta}}$.

(iii) With π_{ref} fixed and the output set finite,

$$\nabla_\theta D_{\mathrm{KL}} = \hat{\mathbb{E}}_s \sum_y \nabla_\theta \pi_\theta(y \mid s) \left[\log \frac{\pi_\theta(y \mid s)}{\pi_{\mathrm{ref}}(y \mid s)} + 1 \right]. \tag{28}$$

Since $\sum_y \pi_\theta(y \mid s) = 1$ identically in θ, we have $\sum_y \nabla_\theta \pi_\theta(y \mid s) = 0$, so the $+1$ term vanishes exactly. Evaluating at $\bar{\theta}$ and writing $\nabla \pi_{\bar{\theta}} = \bar{\pi}\, \nabla \log \pi_{\bar{\theta}}$ gives the stated expression, in which no factor of A appears.

(ii) Setting $\pi_{\mathrm{ref}} = \bar{\pi}$ in (iii) makes the logarithm identically zero. Independently, $g(\theta) = D_{\mathrm{KL}}(\pi_\theta \| \bar{\pi})$ is nonnegative with $g(\bar{\theta}) = 0$ by Gibbs' inequality, so $\bar{\theta}$ is an unconstrained global minimizer of a differentiable function and $\nabla g(\bar{\theta}) = 0$. That the bias in (iii) is generally nonzero is seen on a two-outcome example: with $\pi_\theta(1) = \sigma(\theta)$, $\sigma(\bar{\theta}) = \frac{3}{4}$ and $\pi_{\mathrm{ref}}(1) = \frac{1}{2}$, the derivative is $\frac{3}{16} \log 3 = 0.206 \neq 0$.

(iv) Differentiating the expression in (iii) once more and evaluating at $\bar{\theta}$ with $\pi_{\mathrm{ref}} = \bar{\pi}$, the two terms carrying the factor $\log(\pi_{\bar{\theta}}/\bar{\pi}) = 0$ drop out and what remains is $\hat{\mathbb{E}}_s \mathbb{E}_{y \sim \bar{\pi}}[\nabla \log \pi_\theta\, \nabla \log \pi_\theta^\top] = F(\bar{\theta})$. Since $g(\bar{\theta}) = 0$ and $\nabla g(\bar{\theta}) = 0$, Taylor's theorem gives $D_{\mathrm{KL}}(\pi_{\bar{\theta}+\Delta} \| \bar{\pi}) = \frac{1}{2} \Delta^\top F(\bar{\theta}) \Delta + o(\|\Delta\|^2)$. Since $F(\bar{\theta})$ is positive semidefinite but need not be definite in an overparameterized model, this penalizes distribution-changing directions and leaves Fisher-null directions unpenalized to second order. □

Items (ii) and (iii) separate the two anchorings exactly. Anchoring to the round-start snapshot leaves the first-order direction equal to the group-relative policy gradient, so the regularizer costs nothing at the point where the data was generated. Anchoring to a different reference can add a first-order term that does not depend on the reward and need not vanish, so it can pull the update away from the sampled distribution before any advantage is consulted.

D.5 Group Normalization Cancels Unconditional Costs

Proposition 4. *Let $\mathcal{G}$ be a group of $n \geq 2$ turns with raw scores u, let $b_i = \mathbb{1}[\Delta\phi_i = 0]$, and write $r^{(c)} = u - \lambda c\, b$ for the conditional cost and $\tilde{r}^{(c)} = u - \lambda c\, \mathbf{1}$ for an unconditional one. Then:* (i) *$z(av + \kappa \mathbf{1}) = z(v)$ for every $a > 0$ and $\kappa \in \mathbb{R}$.* (ii) *$z(\tilde{r}^{(c)}) = z(u)$ for every c.* (iii) *If b is constant on $\mathcal{G}$ then $z(r^{(c)}) = z(u)$ for every c.*

(iv) *if b is non-constant, u equals u_1 on $\{b_i = 0\}$ and u_0 on $\{b_i = 1\}$, and $\delta_c = u_1 - u_0 + \lambda c > 0$, then with $\alpha = |\{i : b_i = 1\}|/n$ and the convention $s^2 = n^{-1} \sum_i (\cdot)^2$,*

$$z(r^{(c)})_i = \begin{cases} \sqrt{\alpha/(1-\alpha)}, & b_i = 0, \\ -\sqrt{(1-\alpha)/\alpha}, & b_i = 1, \end{cases} \tag{29}$$

independently of c.

Proof. (i) Let $w = av + \kappa \mathbf{1}$ with $a > 0$. Then $\bar{w} = a\bar{v} + \kappa$ and $\sum_i (w_i - \bar{w})^2 = a^2 \sum_i (v_i - \bar{v})^2$, so $s(w) = a\, s(v)$ under either convention and $s(w) = 0$ exactly when $s(v) = 0$. If $s(v) > 0$ then $z(w)_i = a(v_i - \bar{v})/(a\, s(v)) = z(v)_i$. If $s(v) = 0$ both sides vanish by the normalization convention.

(ii) Apply (i) with $a = 1$ and $\kappa = -\lambda c$: an unconditional cost is a pure shift and is therefore cancelled exactly.

(iii) If $b \equiv b_0$ then $r^{(c)} = u - \lambda c\, b_0 \mathbf{1}$, again a pure shift.

(iv) Here $\alpha \in (0, 1)$ because b is non-constant. The scores are u_1 on progressing turns and $u_1 - \delta_c$ on stalling turns, so $\bar{r}^{(c)} = u_1 - \alpha\delta_c$ and the deviations are $\alpha\delta_c$ and $-(1-\alpha)\delta_c$ respectively. Hence

$$s^2 = (1-\alpha)\, \alpha^2 \delta_c^2 + \alpha\, (1-\alpha)^2 \delta_c^2 = \delta_c^2\, \alpha(1-\alpha), \tag{30}$$

so $s = \delta_c\sqrt{\alpha(1-\alpha)} > 0$, and dividing the two deviations by s gives the stated values, in which δ_c and hence c have cancelled. Under the $(n-1)$ convention every entry is multiplied by $\sqrt{(n-1)/n}$, still free of c. □

Part (ii) is the precise form of the claim that an unconditional constant is cancelled by group normalization, which is why the cost is charged conditionally. Parts (iii) and (iv) show that the conditional form has no effect when the group does not mix stalling and progressing turns, and that its effect does not vary with c when raw scores are constant within each of the two classes and $\delta_c > 0$. Part (iv) needs $\delta_c > 0$, which holds for non-terminal turns: a stalling non-terminal turn has raw score 0, while a progressing turn scores at least $\lambda\Delta\phi > 0$. Without that condition the conclusion fails, as $n = 2$ with $\lambda = 1$, a stalling score of 2 and a progressing score of 0 shows: there z flips sign as c passes 2. A group of three turns with $\lambda = 1$, $b = (1, 0, 0)$ and $u = (0, 1, 2)$ shows that the conditional cost can change advantages, since z moves from $(-1.225, 0, 1.225)$ at $c = 0$ to $(-1.336, 0.267, 1.069)$ at $c = 1$.

D.6 The Repetition Penalty Rules Out Point-Mass Optima

We treat one environment and let $m_s = n_s/N$ be the batch frequency of signature s among its N proposals. Under the continuous allocation relaxation m ranges over the whole simplex, that is, the integrality constraint $Nm_s \in \mathbb{Z}_{\geq 0}$ is dropped. Under realizability the Challenger can induce any such m while the conditional task distribution Q_s within each signature stays fixed.

Proposition 5 (No signature collapse)**.** *Let $\mathcal{S}$ be the finite set of reachable difficulty signatures with $L = |\mathcal{S}| \geq 2$, and let $f_s = \mathbb{E}_{x\sim Q_s}[\min(p(x), 1-p(x))] \in [0, \frac{1}{2}]$, the expectation taken over task generation and Solver rollouts. Under the frozen snapshot, the continuous relaxation and realizability, the objective*

$$J_N(m) = \sum_s m_s\big(f_s - \rho\big) = \sum_s m_s f_s - \sum_s m_s^2 + \tfrac{1}{N} \tag{31}$$

has a unique maximizer $m^\star$, given by $m_s^\star = \frac{1}{2}(f_s - \eta)_+$ where η is the unique real with $\sum_s (f_s - \eta)_+ = 2$. Writing $S^\star = \{s : m_s^\star > 0\}$, $k = |S^\star|$ and $\bar{f} = k^{-1}\sum_{s\in S^\star} f_s$,

$$m_s^\star = \frac{1}{k} + \frac{f_s - \bar{f}}{2}, \qquad s \in S^\star, \tag{32}$$

and consequently $k \geq 2$, so no point mass is optimal, with $|m_s^\star - m_{s'}^\star| \leq \frac{1}{4}$ for all $s, s' \in S^\star$ and $m_{\max}^\star \leq \frac{1}{k} + \frac{1}{4}$.

Proof. The penalty depends on x only through $\mathrm{sig}(x)$, and an emission whose signature has frequency m_s carries $\rho = (Nm_s - 1)/N = m_s - 1/N$. Conditioning on the signature and applying the tower rule gives Eq. 31. The additive $1/N$ does not affect the maximizer, so it suffices to maximize $J(m) = \langle m, f\rangle - \|m\|_2^2$ over the simplex.

$\nabla^2 J = -2I_L \prec 0$, so J is strictly concave, and the relaxed simplex is nonempty, compact and convex. A continuous function on a compact set attains its maximum. If $m^{(1)} \neq m^{(2)}$ were both maximizers then strict concavity would give $J(\frac{1}{2}m^{(1)} + \frac{1}{2}m^{(2)}) > \max J$ at a feasible point, which is impossible.

The problem is concave with affine constraints, so the Karush-Kuhn-Tucker conditions are both necessary, by the linearity constraint qualification, and sufficient. With multiplier η for $\sum_s m_s = 1$ and $\mu_s \geq 0$ for $m_s \geq 0$, stationarity reads $f_s - 2m_s - \eta + \mu_s = 0$ with $\mu_s m_s = 0$. If $m_s > 0$ then $\mu_s = 0$, so $m_s = \frac{1}{2}(f_s - \eta)$ and $f_s > \eta$. If $m_s = 0$ then $\mu_s = \eta - f_s \geq 0$, so $f_s \leq \eta$ and $\frac{1}{2}(f_s - \eta)_+ = 0 = m_s$. Both cases give $m_s = \frac{1}{2}(f_s - \eta)_+$, and feasibility forces $\sum_s (f_s - \eta)_+ = 2$. The map $\Psi(\eta) = \sum_s (f_s - \eta)_+$ is continuous and non-increasing, equals 0 for $\eta \geq \max_s f_s$, tends to $+\infty$ as $\eta \to -\infty$, and is strictly decreasing wherever it is positive, because at least one summand is then positive and strictly decreasing there. So the level $\Psi(\eta) = 2 > 0$ is attained at exactly one η.

The support is nonempty since the masses sum to one. Summing $m_s^\star = \frac{1}{2}(f_s - \eta)$ over $S^\star$ gives $2 = k\bar{f} - k\eta$, so $\eta = \bar{f} - 2/k$, and substituting back yields Eq. 32.

Suppose $k = 1$, say $S^\star = \{s_0\}$. Then $\bar{f} = f_{s_0}$ and $\eta = \bar{f} - 2/k = f_{s_0} - 2$. Since $L \geq 2$ there is some $s_1 \neq s_0$, and $s_1 \notin S^\star$ requires $f_{s_1} \leq \eta = f_{s_0} - 2$. But $f_{s_1} \geq 0$ and $f_{s_0} \leq \frac{1}{2}$, so $0 \leq f_{s_1} \leq \frac{1}{2} - 2 = -\frac{3}{2}$, a contradiction. Hence $k \geq 2$.

For $s, s' \in S^\star$, Eq. 32 gives $m^\star_s - m^\star_{s'} = \frac{1}{2}(f_s - f_{s'})$, and f takes values in $[0, \frac{1}{2}]$, so the difference is at most $\frac{1}{4}$ in absolute value. Likewise $\max_{S^\star} f - \bar{f} \leq \frac{1}{2}$, so $m^\star_{\max} = \frac{1}{k} + \frac{1}{2}(\max_{S^\star} f - \bar{f}) \leq \frac{1}{k} + \frac{1}{4}$. □

The bound is uniform in the Solver's competence profile: whatever capture rates the frontier values f_s encode, the optimal allocation spreads over at least two signatures and no signature exceeds another by more than $\frac{1}{4}$.