TRAINING RECURSIVE AGENT SWARMS AT DEPTH

Anonymous authors
Paper under double-blind review

ABSTRACT

Agent swarms tackle complex tasks through recursive delegation, where workers can delegate further. However, upstream learning by parents and other ancestors can reduce delegation before independently updated workers receive enough training to contribute. We model this dependence through update coverage. For a policy, this is the probability that a root-task rollout group provides usable training data. Passive sampling costs can grow exponentially with delegation depth. In a finite-batch model, individually beneficial upstream updates can jointly reduce downstream coverage and lower two-stage team value after the child's subsequent learning, even under their best order. We introduce LACA (Learning Aware Controlled Adaptation) to supply nested policies with usable training data. LACA restores saved parent states at target child calls, samples fresh continuations, and selects ordered policy subsets using coverage and estimated two-stage team value. It retains the original task reward and learner objective. We compare paired update schedules, test direct child entry, and restore workers to pre-RL policies. Across WebWalkerQA, Terminal-Bench 2.1, and BrowseComp-Plus, LACA improves mean task success over its GRPO and DAPO inner learners by 6.14 and 5.57 percentage points, respectively. Code is available online.

1 INTRODUCTION

Agent swarms offer a promising way to solve complex tasks through delegation and coordinated reasoning (Wu et al., 2024; Zhuge et al., 2024; Lan et al., 2026). Agent coevolution takes this idea further by allowing collaborating policies to learn from team rewards (Xu et al., 2026; Hong et al., 2025). For example, a worker diagnosing a software fault may delegate construction of a minimal reproduction. Nested delegation lets workers break down subproblems that remain difficult after the first split. When policies learn independently, upstream agents (a worker's parent and other ancestors) also control its calls and training opportunities. *How can an agent swarm learn together when upstream updates can reduce subagents' training opportunities before their capabilities develop?*

Natural sampling reaches workers through ancestor delegation; relying on undertrained workers may lower team rewards. Within a GRPO group, below-average trajectories receive negative advantages, while uniform-reward groups provide no relative signal. Figure 1(b) shows an S_1 update that favors returning without further delegation: C's policy is fixed, but receives fewer calls and usable groups. Saved parent states in panel (a) enable controlled entry. Feng et al. (2026) relate role instability to advantage normalization, while Zeng et al. (2026) relate role gradients to workflow structure.

For each policy, update coverage is the probability that a root-task rollout group yields a completed call, usable response tokens, and a nonzero learning signal. Our analysis gives two findings. First, passive sampling costs can grow exponentially with delegation depth. Second, in a finite batch model, each upstream update can improve team value when added separately to the same baseline that updates only the child. Yet retaining all of them can reduce that value even in the best order. Reordering can protect the child's current update but cannot remove the cost to its next learning stage. Subset selection differs from update ordering.

We introduce Learning Aware Controlled Adaptation (LACA) to control training opportunities across delegation depth. It restores saved parent states at target child calls and samples fresh continuations, bypassing ancestor routing. LACA selects policy subsets and their order using coverage and estimated two-stage team value after subsequent downstream learning. Selected policies learn sequentially from their own response tokens, with fresh groups collected after each update to reflect changes in their

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.

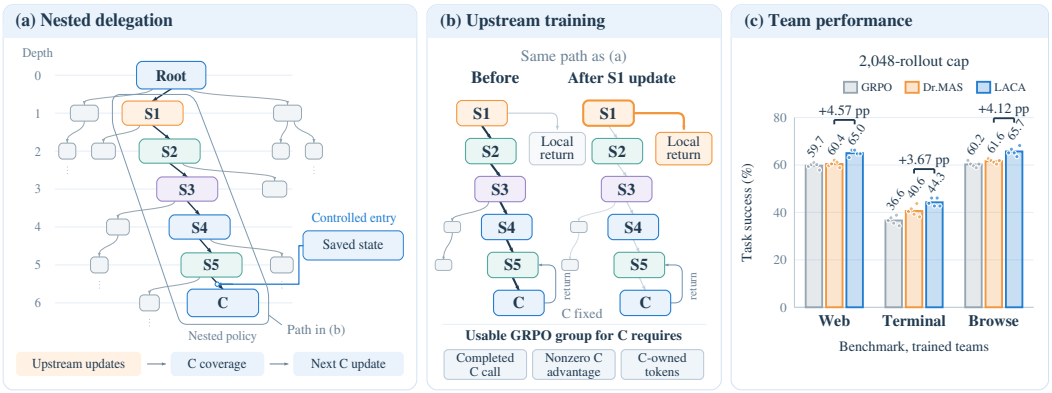

Figure 1: **Nested delegation, usable signal, and team performance.** **(a)** Nested delegation and controlled entry from a saved parent state. **(b)** An illustrative S_1 update reduces calls to a fixed C; usable GRPO groups require completion, nonzero C advantages, and response tokens generated by C. **(c)** Task success for GRPO, Dr.MAS, and LACA (GRPO).

partners. LACA retains the task reward and learner objective. A paired audit evaluates the completed team before replacing the active team.

Controlled studies on COMMAQA-N (Khot et al., 2022) examine whether nested policies remain active and learn, and how their partners affect team gains. Broader evaluation covers web traversal, terminal execution, and retrieval. We initialize a 9B Qwen3.5 root and 4B Qwen3.5 workers (Qwen Team, 2026) with supervised fine-tuning (SFT) on delegation, execution, and recovery examples, then train through environment interaction on TMax-15K (Ivison et al., 2026). LACA improves mean success over GRPO (Shao et al., 2024) and DAPO (Yu et al., 2025) by 6.14 and 5.57 percentage points, respectively. Both LACA variants exceed Dr.MAS on all three benchmarks. Controlled entry–selection comparisons and paired program tests examine component effects, while rollback measures the contribution of learned workers with their partners held fixed.

The main contributions of this work are:

- We formalize update coverage and characterize how depth and coupled updates affect nested learning (Section 4).
- We introduce LACA, combining controlled entry, two-stage selection of policy subsets and their order, and a paired audit of the completed team (Section 5).
- We measure partner-dependent gains and depth-specific signal, and evaluate program selection and final-team dependence with paired tests (Section 6).

2 Related Work

Appendix A gives further discussion.

Agentic frameworks. Multi-agent frameworks organize cooperation through conversations, specialized roles, and computational graphs (Wu et al., 2024; Li et al., 2023; Hong et al., 2024; Zhuge et al., 2024), while lead agents recruit and schedule workers for delegated tasks (Anthropic, n.d.; Zhang, 2026). Inference-time computation supports extended reasoning, repeated sampling, search, and multi-agent aggregation or debate (Wei et al., 2022; Wang et al., 2023b; Yao et al., 2023; Wang et al., 2025; Du et al., 2024). Nested delegation lets workers further decompose their subproblems. With independently trained policies, routing also determines access to training experience.

Agentic learning algorithms. PPO uses critic-based advantages, GRPO compares outcomes within rollout groups, and DAPO filters groups with uniform rewards (Schulman et al., 2017; Shao et al., 2024; Yu et al., 2025). Team training further specifies trainable policies and update schedules. Delegation can be trained with fixed workers (Cui et al., 2026); Kimi K2.5 trains an orchestrator with frozen workers and an auxiliary reward for subagent creation (Kimi Team, 2026). Shared-policy

approaches train main and subagent trajectories at one delegation level in SearchSwarm (Lan et al., 2026), parallel roles in WideSeek-R1, or recursive calls with depth-frequency weights in RAO (Xu et al., 2026; Gandhi et al., 2026). M-GRPO trains distinct main and subagent models and aligns their variable numbers of calls (Hong et al., 2025), while MAGRPO jointly trains collaborating policies with centralized group-relative advantages (Liu et al., 2025). Sequential multi-agent methods organize updates to a given set of agents (Kuba et al., 2022; Wang et al., 2023a). SAT and TeamTR provide sequential update guarantees, with TeamTR resampling after each update (Xie et al., 2026b;a). LACA selects policy subsets and their order using coverage and estimated two-stage team value, accounting for how current updates change subsequent training opportunities.

Credit assignment in agentic learning. Cross-agent credit methods construct role-specific signals from team outcomes (Li et al., 2026a;b; Ben-Gal & Tong, 2026), and C3 evaluates counterfactual continuations from restored histories to estimate per-decision advantages (Chen et al., 2026). Dr.MAS rescales advantages across roles (Feng et al., 2026), while workflow topology affects role gradients (Zeng et al., 2026). Zero-advantage probability has also been analyzed for a single policy (Nie et al., 2026). LACA measures update coverage under the chosen credit rule and uses controlled entry to supply usable training signals to nested policies.

3 Problem Setting

Delegation and policy ownership. At decision stage t, the delegation graph $\mathsf{G}_t = (\mathcal{N}_t, E_t)$ contains runtime agents $\mathcal{N}_t$ and delegation edges E_t. The map $b_t : \mathcal{N}_t \to \mathcal{R}_t$ assigns each agent to an active policy owner in $\mathcal{R}_t$; calls sharing an owner reuse its parameters and optimizer. Our implementation assigns each node an independent owner with a private low-rank adapter (LoRA) on first update; workers share a frozen backbone. Figure 2(a) shows three independent owners on a parser-repair path: coordinator $O \to$ debugger $S \to$ patch specialist C. The state Θ collects adapters and optimizer states; $\mathcal{F}_t$ adds the graph and remaining rollout budget. SOLVE (shown as EXEC in the architecture) executes local work; DELEGATE invokes an existing subagent; CREATE starts a child call assigned an owner by b_t. Child returns resume their waiting parents, and execution ends when the root returns its final answer.

Training groups and usable signal. A root group contains rollouts for one task and entry; each rollout is a team trajectory through child calls and returns to the final root-task reward. Within a fixed-policy window, $n = 1, \ldots, N$ indexes groups under the same task and entry distribution. The indicator $Z_{n,i} = 1$ records usable data for owner i: group-relative training requires i to be called ($A_{n,i}$), complete its call ($F_{n,i}$), receive nonzero relative advantage ($V_{n,i}$), and contribute usable owned response tokens ($W_{n,i}$). Its probability is

$$s_i(\Theta) = \Pr_{\Theta}(Z_{n,i} = 1) = \rho_i f_i v_i \nu_i.$$

Coverage s_i averages over task and trajectory randomness for any group n. The factors ρ_i, f_i, v_i, ν_i give reach, completion, signal, and ownership probabilities, conditioned in that order. Group-relative signal requires reward variation. Owned responses train i; prompts, tool outputs, and other owners' returns are context. Supervised fine-tuning (SFT) and on-policy distillation (OPD) instead require an owner-local target; PPO requires nonzero critic-based actor advantage, counting critic-only updates separately. The effective count $N_i^{\text{eff}} = \sum_{n=1}^{N} Z_{n,i}$ counts each group once per owner. Each update has a budget of B groups and a predeclared effective-group requirement m_i. These counts measure groups; the rollout budget charges individual sampled trajectories.

A depth-d site has reach $\prod_{h=1}^{d} p_h$, where p_h is the probability of reaching level h given level $h-1$. Site coverage counts usable groups at each site; owner coverage pools all its sites. Natural entry starts at the root; controlled entry samples fresh continuations from saved parent states to the root reward.

Candidate policies and audit evidence. Let c denote the downstream owner under study, active or proposed. A proposed $c \notin \mathcal{R}_t$ has a fixed parent, interface, and audit protocol and is tested in a temporary branch before admission. Conditional on candidate-call completion, $Y_{n,c}$ is the paired difference of candidate/control group-mean final rewards. Audit exposure s_c^{ev} is the probability of observing this paired score; training coverage s_c also requires learner signal and owned tokens. Team capability Q uses fixed inference limits; Q^{aud} evaluates the complete team on root audit tasks.

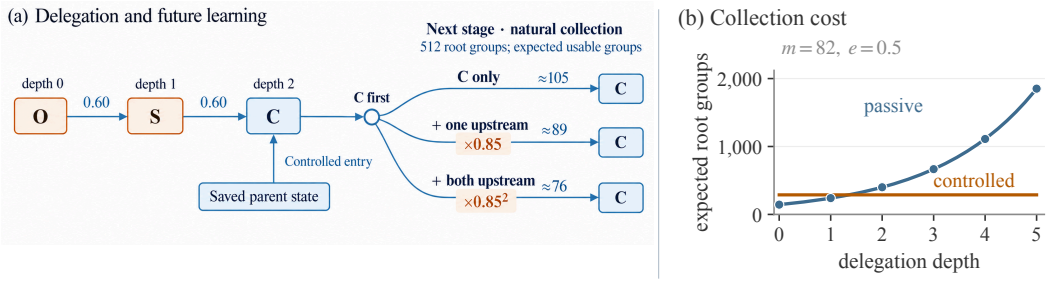

Figure 2: **A theoretical example of depth and update coupling.** **(a)** Delegation and expected effective C groups under natural collection; controlled entry resumes a saved parent state. **(b)** Expected root groups for $m = m_c = 82$ effective groups; controlled exposure is $e = e_c = 0.5$.

Co-evolving updates and team value. An update Δ_j can change owner j's actions and returns, shifting the calls, contexts, and usable groups available to i:

$$\Delta^{\mathrm{cov}}_{j\to i} = s_i(\Theta + \Delta_j) - s_i(\Theta).$$

A positive shift increases i's coverage. A program $u = (\mathcal{A}, \sigma)$ updates owner subset $\mathcal{A}$ in order σ; collection cost depends on the number of updates. For each non-STOP program, continuation ω_t updates c next stage with upstream policies frozen, giving state $\mathcal{F}_{t+2}^{u,\omega_t}$. The STOP reference applies no policy update in either stage. The expected two-stage team gain is

$$\Gamma_t^{(2)}(u;\omega_t) = \mathbb{E}\big[Q^{\mathrm{aud}}(\mathcal{F}_{t+2}^{u,\omega_t}) - Q^{\mathrm{aud}}(\mathcal{F}_{t+2}^{\mathrm{STOP},\omega_t}) \,\big|\, \mathcal{F}_t\big], \qquad \Gamma_t^{(2)}(\mathrm{STOP};\omega_t) = 0. \tag{1}$$

The two-stage value measures team gain after further downstream learning; relative advantages determine token-level updates within each stage. Direct gain g_j is owner j's two-stage contribution, excluding changes through c's coverage. Comparing with the c-only program isolates upstream additions. Table 2 summarizes notation; Appendices B.1 and B.1.1 specify task coordinates and assumptions underlying the coverage and value calculations.

4 Training Signal in Co-Evolving Systems

4.1 Training Signal at Depth

In Figure 2(a), the declared call site of $c = C$ is reached through two group-level routing factors of 0.6. Set completion and ownership factors to one and write $m = m_c$. Each activated C group contains eight independent binary-reward rollouts with success probability 0.1, giving mixed-reward probability $v = v_c = 1 - 0.1^8 - 0.9^8$ and coverage $s = s_c = 0.6^2 v \approx 0.205$.

Assumption 1 (Passive and controlled evidence)**.** Within one fixed-policy window, root groups are independent. Each group is effective for policy c with probability s_c and exposes a group-level audit observation $Y_{n,c}$ with probability $s_c^{\mathrm{ev}} \geq s_c$. Conditional on the exposure indicators, observations are independent with mean μ_c and centered sub-Gaussian proxy σ_c^2. Under passive routing, $s_c = (\prod_{h=1}^d p_h) f_c v_c \nu_c$ and $s_c^{\mathrm{ev}} = (\prod_{h=1}^d p_h) f_c$. Controlled activation at the declared parent replaces the path product by a group exposure probability $e_c \in (0, 1]$, keeping the conditional factors and the exposed-observation mean and proxy fixed.

Theorem 1 (Path-compounded evidence and controlled recovery)**.** *Under Assumption 1, the expected number of root groups needed to collect m effective groups is exactly m/s_c. The root-group requirement for identifying the exposed mean to error ϵ scales with $1/s_c^{\mathrm{ev}}$, with upper and matching lower bounds in Appendices B.2 and B.2.1. Under controlled activation, the same statements hold with $s_c^{\mathrm{ctl}} = e_c f_c v_c \nu_c$ and $s_c^{\mathrm{ev,ctl}} = e_c f_c$, which remove the upstream path product.*

Corollary 1 (Depth dependence and critical depth)**.** *For a depth-d candidate, let s_c^{pass} denote its passive update coverage in Assumption 1. If $p_h \leq \bar{p} < 1$ for every edge and $f_c v_c \nu_c$ is bounded above, then both the training requirement m/s_c^{pass} and the identification requirement of Theorem 1 are $\Omega(\bar{p}^{-d})$. Passive identification at a declared precision therefore becomes infeasible beyond a critical depth fixed by the remaining budget (Appendix B.2.2).*

Effective counts are binomial; waiting for m follows a negative binomial. Figure 2(b) evaluates the cost $m/(0.6^d v)$ for the same example. At $d = 2$ and $m = 82$, passive collection requires about 400 root groups in expectation; controlled entry with $e_c = 0.5$ requires about 288. Completion, signal availability, and owned tokens still determine whether an exposed group is usable.

Result 1: passive collection costs can grow exponentially with depth.

4.2 Coupled Updates and Update Order

Upstream updates to O and S can improve team performance while reducing C's usable groups.

Assumption 2 (Finite-batch effective update)**.** Each of the B root groups is effective for downstream policy c independently with probability s_c; the update gains $G_c > 0$ when at least m_c effective groups arrive and $\underline{G}_c < G_c$ otherwise. An upstream update j gains $g_j > 0$ directly. After a set J of upstream commits, coverage is $s_c\Lambda_J$, where Λ_J cannot increase as more upstream commits enter J, with $\Lambda_{\{j\}} = \lambda_j \in (0, 1]$ (full two-stage bookkeeping in Appendix B.1.1).

The probability of meeting the requirement is $\Psi_{B,m}(s) = \Pr\{\text{Binomial}(B, s) \geq m\}$, giving expected gain $\mathcal{G}_{B,m}(s) = \underline{G}_c + (G_c - \underline{G}_c)\Psi_{B,m}(s)$ at coverage s. The binomial tail transitions near m/B and sharpens when $m = m_B$ grows with B such that $m_B/B \to \tau \in (0, 1)$.

For the same example, set $B = 512$, $m = 82$, $G_c = 10$, and $\underline{G}_c = 0$. Each upstream update contributes two-stage direct gain $g_j = 2$ and multiplies coverage by 0.85; one and both updates give coverage 0.174 and 0.148, on opposite sides of $m/B \approx 0.160$. Exact binomial averaging yields two-stage values of 19.9 for C alone, 20.1 for either single addition with C first, 16.3 for both in their best order, and 8.8 top-down. Even with C first, both upstream commits incur a next-stage learning loss of 7.57, exceeding their direct gain of 4. Appendix B.3 gives the calculation.

Theorem 2 (Individually useful updates can fail in every order)**.** *Under Assumption 2, let J be a set of upstream updates and write Λ_J for its joint coverage multiplier. If*

$$\begin{aligned} g_j &> \underbrace{\mathcal{G}_{B,m}(s_c) - \mathcal{G}_{B,m}(s_c\lambda_j)}_{\textit{price of one commit}} \quad \textit{for every } j \in J, \\ \sum_{j\in J} g_j &< \underbrace{\mathcal{G}_{B,m}(s_c) - \mathcal{G}_{B,m}(s_c\Lambda_J)}_{\textit{joint coverage response}}, \end{aligned} \tag{2}$$

then every upstream update improves two-stage team value when added separately to the same c-only baseline in c-first order. Retaining all updates is strictly worse than that baseline in every order.

Ordering changes the current-stage coverage cost; subset selection also changes the cost carried into the next stage. We suppress fixed t and ω_t in $\Gamma^{(2)}(u)$.

Corollary 2 (Current and continuation coverage costs)**.** *Under Assumption 2, set $C(J) = \mathcal{G}_{B,m}(s_c) - \mathcal{G}_{B,m}(s_c\Lambda_J)$, with $\Lambda_\emptyset = 1$. Compare u_c, which updates only c, with u, which also updates J, with J_{pre} preceding c:*

$$\Gamma^{(2)}(u) - \Gamma^{(2)}(u_c) = \sum_{j\in J} g_j - C(J) - C(J_{\text{pre}}).$$

The next-stage cost is $C(J)$; the current-stage order cost is $C(J_{\text{pre}})$. Placing c first maximizes value by removing the latter; every order retains $C(J)$.

Result 2: fewer updates can outperform every full order.

Coverage sets the arrival rate of usable groups; loss weights set update direction and size (Appendix B.7), complementing signal rescaling (Feng et al., 2026) and role-collapse analysis (Zeng et al., 2026). Appendices B.4 and B.4.1 give proofs. Proposition 1 gives a nonempty finite-budget region in coverage margins and direct gains. Appendix B.5 analyzes parent-state counts and low-coverage tails; Appendix B.6 proves every-order failure at large budgets for quadratic updates averaged over usable contributions, without a count threshold. These results motivate controlled entry to supply usable groups to nested policies and program selection to account for how upstream changes affect their next learning stage.

5 Learning Aware Controlled Adaptation

Coverage and budget requirements determine program feasibility; estimated two-stage team gain ranks feasible programs; a paired audit determines commitment (Figure 3). Temporary copies of the active team evaluate each nonempty candidate with a common downstream learning step, valuing the training opportunities it leaves. The selected non-STOP program is re-executed from the active team; only the current program is committed. Algorithm 1 in Appendix C.1 gives the procedure.

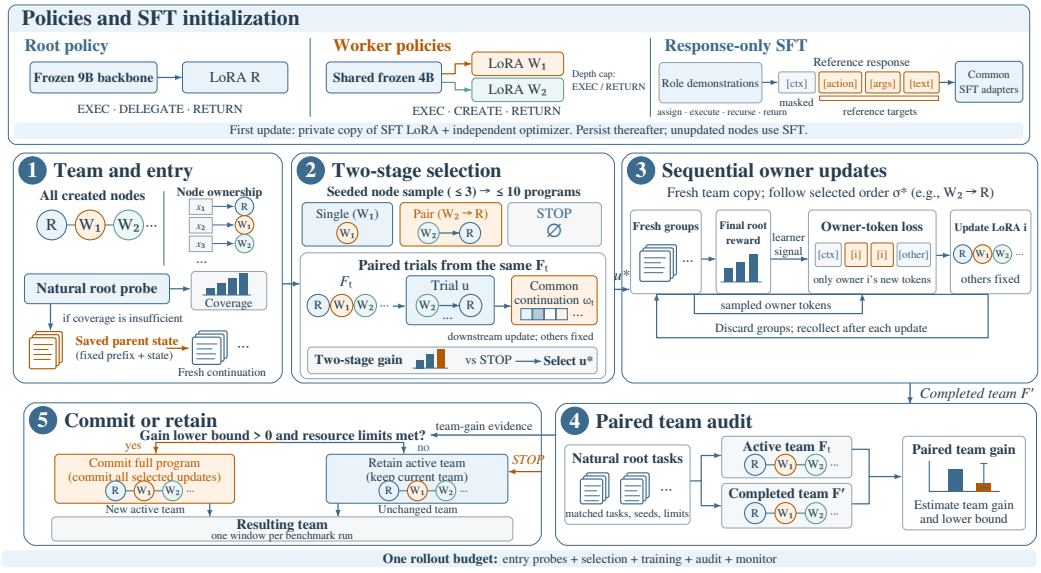

Figure 3: **Training architecture of LACA.** SFT provides common adapters; RL updates private node LoRAs using final root rewards. Backbones remain frozen; GRPO or DAPO supplies the inner learner. Step 2 samples nodes for lookahead; steps 3–5 execute, audit, and commit the current program.

Curriculum for controlled entry. LACA probes owner coverage and selects saved parent states when natural collection cannot meet effective-group requirements within budget. Training entries restore saved parent contexts and the associated execution state; terminal entries replay the saved prefix in a fresh environment. LACA uses restored-history continuation (Chen et al., 2026), fixing the task, owner, environment, parent contexts, returns, and remaining budget. Each continuation samples fresh responses under the current adapters. Child returns resume the waiting parents until the root answers and receives the task reward. Saved responses remain context; only new owner responses enter the loss. Evaluation starts at the root. Appendix C.1.2 describes candidates and bootstrap inputs.

Program selection and owner updates. Theorem 2 and Corollary 2 motivate subset and order selection over a declared finite set of ordered owner subsets and STOP. Paired development trials estimate coverage and two-stage gain against STOP; the common continuation updates downstream c with upstream owners frozen (Appendix D.4). Coverage guides entry and collection. LACA ranks completed, budget-feasible programs by mean two-stage development gain. Ties favor shorter programs, with STOP winning zero-gain ties. The active team is retained when $u^\star = \text{STOP}$.

Otherwise, $u^\star = (\mathcal{A}^\star, \sigma^\star)$ runs on a fresh copy $\mathcal{F}'_t = \mathcal{F}_t$. Each owner collects until it reaches its effective-group requirement or budget. If the requirement is unmet, LACA discards the whole temporary program. Under b_t, only the scheduled owner's response tokens enter the original learner objective with final-team-reward advantages; prefixes and other owners' returns are context. Only its adapter and optimizer change; backbones and other owners remain fixed. Used groups are discarded; each subsequent owner receives fresh groups under the updated $\mathcal{F}'_t$.

Safe utility gate. The audit pairs completed $\mathcal{F}'_t$ with the active team on the same root tasks, seeds, and inference limits. Commitment requires all selected updates to complete and a positive one-sided 95% paired-bootstrap lower bound on team gain. Token, action, and depth limits apply throughout. Otherwise, the active team is retained. Appendix D.4 gives the benchmark gate; Appendix C.2.4 analyzes sufficient conditions for a coverage-certified controller. Entry, selection, training, audit, and

monitor executions share one budget, including rejected programs. Each benchmark run uses one decision window and evaluates the resulting team from the root.

6 EXPERIMENTS

6.1 SETUP AND MAIN RESULTS

Harness and initialization. Following SearchSwarm's delegation-and-return pattern (Lan et al., 2026), we use serial recursive calls with Harbor terminal execution. All methods share response-only SFT of a Qwen3.5-9B root and Qwen3.5-4B workers (Qwen Team, 2026) on 928 targets (209 root; 719 worker), teaching delegation, tool use, return, and parent resumption. Each node is an independent update owner on frozen backbones; workers share the 4B backbone. Its first update copies the common SFT adapter into a private LoRA and creates a separate optimizer state. Later updates reuse both; unupdated nodes retain SFT. Actor losses use final team rewards and only the updated node's response tokens. Inference fixes parameters: nodes execute, delegate, or return, and parents resume to check and integrate child results.

Training and controlled tests. RL uses TMax-15K (Ivison et al., 2026), with fresh environment trajectories and official verifier outcomes as team rewards. GRPO and DAPO batch eight tasks with 32 trajectories each. LACA collects groups of eight trajectories and updates each owner after four effective groups, within eight raw groups. Main results use a shared 2,048-rollout cap, including controller costs. Root-started evaluation uses WebWalkerQA, Terminal-Bench 2.1, and BrowseComp-Plus. Controlled studies examine COMMAQA-N decompositions, update coverage, program combinations, and component effects under fixed sampling budgets. Appendices D.2.2 and D.3 describe the SFT data and initialization; Appendix D.4 gives the learner settings.

Methods and evaluation. We compare the base checkpoints and frozen SFT with GRPO (Shao et al., 2024), DAPO (Yu et al., 2025), and Dr.MAS (Feng et al., 2026). LACA (GRPO) and LACA (DAPO) apply the controller to their respective learner objectives. Dr.MAS uses its published learner configuration. Separate development tasks assess candidates. Inference allows up to 65,536 generated tokens and 64 actions per team execution, with recursive depth capped at six. Appendices C.6, D.4, and D.5.3 give the complete protocols.

Figure 4 compares the group-relative learners and their LACA variants. LACA improves success over its GRPO and DAPO inner learners on all three benchmarks, averaging 6.14 and 5.57 percentage points, respectively. LACA (DAPO) performs best on Terminal-Bench and BrowseComp-Plus, while LACA (GRPO) performs best on WebWalkerQA. Table 8 also reports the base and frozen SFT teams.

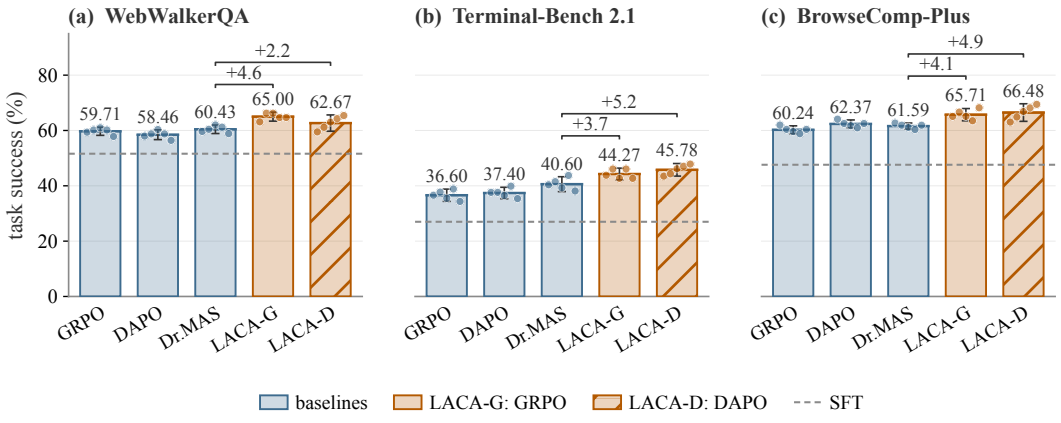

Figure 4: **Task performance under a 2,048-rollout cap.** Bars and labels show mean task success across five runs. Dots show individual runs; whiskers give 95% t confidence intervals. Steel denotes baselines and orange LACA; hatching identifies LACA (DAPO). Dashed lines mark frozen SFT; brackets give each LACA variant minus Dr.MAS (pp).

6.2 Mechanism Analysis

At the common SFT checkpoint, Figure 5 measures usable-group coverage through depth five. Natural entry loses coverage with depth; controlled entry maintains access on all three benchmarks. At depth five, mean coverage rises from 8.75% to 50.94%. With policies fixed, coverage measures the path-level training access in Theorem 1. Appendix E.1 distinguishes site and owner coverage.

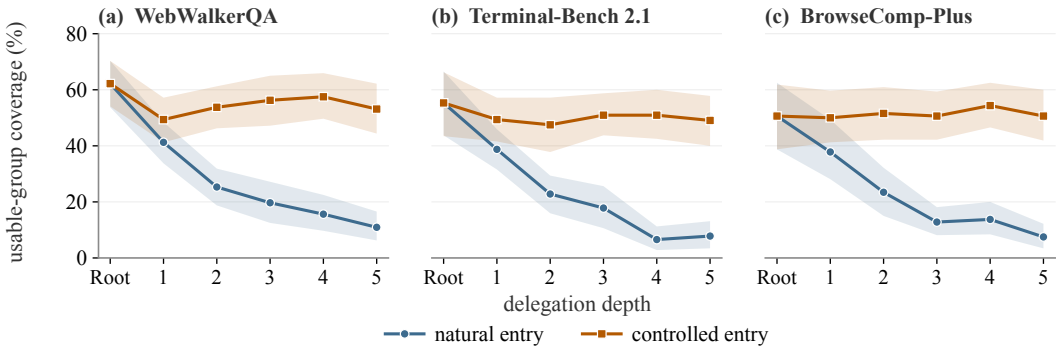

Figure 5: **Controlled entry preserves access to deep training sites.** Target-site coverage from root to depth five at the common SFT checkpoint. Steel circles denote natural entry; orange squares denote controlled entry. Each benchmark has five repeats of 64 groups; bands give 95% hierarchical repeat/group bootstrap intervals.

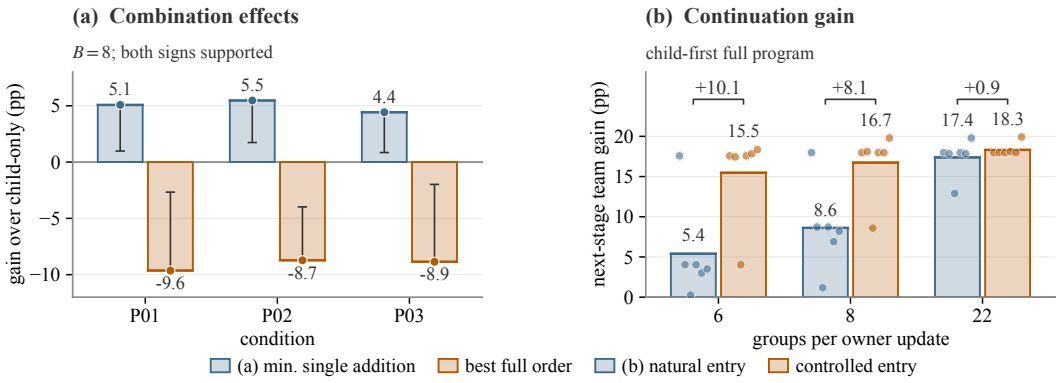

Figure 6: **Combination effects and controlled continuation. (a)** Minimum single-addition and best full-order Q_2 gains over child-only training; whiskers give one-sided simultaneous 95% bounds. **(b)** Continuation gain $Q_2 - Q_1$ after the child-first full program: bars average six teams; dots show team values; brackets give controlled minus natural gain (pp). Figure 14 reports all cells.

Update combinations. We test Theorem 2's update-combination effect by training LLM teams under six conditions at $B = 6, 8, 22$ groups per owner update. Each cell compares final Q_2 after a common child continuation: two single upstream additions and six full orders versus child-only training that updates c in both stages. Figure 6(a) shows combination failure at $B = 8$: minimum single-addition gains range from 4.43 to 5.47 points, and best full-order gains from -9.64 to -8.72 points. The simultaneous bounds support gains for both single additions and losses for all six full orders, motivating subset selection alongside ordering (Appendix E.2.1). Case 1 in Appendix E.6 gives a routing example and the associated child-continuation training counts.

In the program-library experiment (Figure 6(b)), both entries start from the same checkpoint after (c, j_1, j_2), with team value Q_1, and update only c while keeping upstream owners fixed. Controlled entry raises the chance of meeting the effective-group requirement and increases next-stage team gain by 10.07, 8.12, and 0.93 points at $B = 6, 8, 22$, respectively. Natural entry already meets the requirement with high probability at the largest budget, where additional gain is small. Appendix E.3 reports expected effective-group counts and probabilities of meeting m_c.

6.3 Component Effects and Worker Contributions

Table 1: **Component effects and worker contributions under a 2,048-rollout cap.** (A) GRPO task success; bold and underlined values mark the best and second-best scores. (B) Two-stage minus one-stage success in the repeated-window study (controlled entry, gate off). (C) W1/W2 rollback minus full-LACA success. Paired intervals appear in Tables 10, 17, and 7.

Configuration	WebWalkerQA	Terminal-Bench 2.1	BrowseComp-Plus	Mean
(A) Entry and program: task success (%)				
Natural + fixed	36.25	13.26	18.59	22.70
Controlled + fixed	<u>43.28</u>	<u>15.73</u>	<u>25.00</u>	<u>28.00</u>
Natural + selected	41.56	15.28	23.12	26.66
LACA	**65.00**	**44.27**	**65.71**	**58.33**
(B) Two-stage objective: gain (pp), controlled entry and gate off				
Two-stage − one-stage	+2.34	+1.57	+2.03	+1.98
(C) Worker rollback: change (pp) relative to full LACA				
Restore W1	−4.22	−2.70	−4.53	−3.82
Restore W2	−1.09	−0.45	−0.94	−0.83
Restore W1 + W2	−5.16	−3.37	−5.78	−4.77

Component ablation. Table 1 compares entry and program control in (A) and the selection objective in (B). With the fixed three-node program, controlled entry adds 5.30 points. Under controlled entry, LACA's program controller adds 30.32 points over this reference. In the separate repeated-window study, with entry controlled and the gate off, the two-stage objective improves mean success over the one-stage objective by 1.98 points. With the two-stage objective fixed, enabling the gate adds 27.48 points. This study's monitor (Appendix E.4) records 3 rejects in 20 decisions, including 1 scored positive by the independent monitor, and commits none scored negative. Without the gate, the one-stage and two-stage configurations commit 5 and 3 negative-scored updates. Full factorial, objective, and program-space results appear in Table 10 and Appendices E.4 and E.2.2.

Downstream-owner dependence. Table 1(C) tests whether the final team relies on its learned workers. Restoring W1 and W2 to SFT with all other nodes fixed costs 4.77 points on average. W1 accounts for most of this loss, while the smaller W2 effect has intervals including zero. This intervention measures the value of learned worker responses in the trained team, including how the unchanged parent uses those responses. A complementary training comparison holds the one-stage, no-gate controller fixed: training workers has mixed effects at early budgets and improves all three benchmarks at 2,048 rollouts, averaging 4.50 points over root-only training. The training and rollback comparisons in Appendices D.5.2 and E.5 show that the final team benefits from worker adaptation under the shared inference limits.

7 Conclusion

Learning in nested agent teams depends on access to usable training data, and that access changes as partners learn. Update coverage makes this dependence explicit: delegation paths can compound collection costs, while individually beneficial upstream updates can jointly reduce the value of a downstream policy's next learning stage. The two-stage analysis separates the cost that ordering can remove from the cost of retaining upstream changes. LACA addresses training access and update interactions through controlled entry and selection of ordered policy subsets, retaining the task reward and learner objective. Across three benchmarks, LACA improves mean success over its GRPO and DAPO inner learners by 6.14 and 5.57 percentage points. Controlled studies examine training access, program choice, and worker adaptation, supporting evaluation of a team update by both its immediate benefit and the learning opportunities it leaves for its partners.

REFERENCES

Anthropic. Orchestrate teams of Claude Code sessions. https://code.claude.com/docs/en/agent-teams, n.d. Accessed September 22, 2026.

Elai Ben-Gal and Stela Tong. Marginal-contribution policy gradients under filtered feedback for multi-agent LLMs. *arXiv preprint arXiv:2604.22785v2*, 2026.

Yanjun Chen, Yirong Sun, Hanlin Wang, Jinghan Wang, Xinming Zhang, Xiaoyu Shen, et al. Exact is easier: Credit assignment for cooperative LLM agents. *arXiv preprint arXiv:2603.06859*, 2026.

Brian Cho, Ana-Roxana Pop, Ariel Evnine, and Nathan Kallus. SNPL: Simultaneous policy learning and evaluation for safe multi-objective policy improvement. *arXiv preprint arXiv:2503.12760*, 2025.

Zhiqing Cui, Haotong Xie, Jiahao Yuan, Cheng Yang, Hanqing Wang, Yuxin Wu, et al. Uno-Orchestra: Parsimonious agent routing via selective delegation. *arXiv preprint arXiv:2605.05007*, 2026.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, 2024. arXiv:2305.14325.

Lang Feng, Longtao Zheng, Shuo He, Fuxiang Zhang, and Bo An. Dr. MAS: Stable reinforcement learning for multi-agent LLM systems. *arXiv preprint arXiv:2602.08847*, 2026.

Apurva Gandhi, Satyaki Chakraborty, Xiangjun Wang, Aviral Kumar, and Graham Neubig. Recursive agent optimization. *arXiv preprint arXiv:2605.06639*, 2026.

Haoyang Hong, Jiajun Yin, Yuan Wang, Jingnan Liu, Zhe Chen, Ailing Yu, Ji Li, Zhiling Ye, Hansong Xiao, Yefei Chen, Hualei Zhou, Yun Yue, Minghui Yang, Chunxiao Guo, Junwei Liu, Peng Wei, and Jinjie Gu. Multi-agent deep research: Training multi-agent systems with M-GRPO. *arXiv preprint arXiv:2511.13288*, 2025.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. arXiv:2308.00352.

Hamish Ivison, Junjie Oscar Yin, Rulin Shao, Teng Xiao, Nathan Lambert, and Hannaneh Hajishirzi. TMax: A simple recipe for terminal agents, 2026. URL https://arxiv.org/abs/2606.23321.

Tushar Khot, Kyle Richardson, Daniel Khashabi, and Ashish Sabharwal. Hey AI, can you solve complex tasks by talking to agents? In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 1808–1823, 2022. doi: 10.18653/v1/2022.findings-acl.142.

Kimi Team. Kimi K2.5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. Trust region policy optimisation in multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2022.

Xiaochong Lan, Quan Chen, Kun Tao, Xinyu Tang, Tianshu Wang, Qianggang Cao, Xinyu Kong, Zujie Wen, Zhiqiang Zhang, and Jun Zhou. SearchSwarm: Towards delegation intelligence in agentic LLMs for long-horizon deep research. *arXiv preprint arXiv:2606.09730*, 2026.

Romain Laroche, Paul Trichelair, and Rémi Tachet des Combes. Safe policy improvement with baseline bootstrapping. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: Communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*, 2023. arXiv:2303.17760.

Yanming Li, Xuelin Zhang, Wenjie Lu, Ziye Tang, Maodong Wu, Haotian Luo, et al. Who deserves the reward? SHARP: Shapley credit-based optimization for multi-agent system. *arXiv preprint arXiv:2602.08335*, 2026a.

Zhongyi Li, Wan Tian, Jinju Chen, Huiming Zhang, Yang Liu, Yikun Ban, and Fuzhen Zhuang. Counterfactual credit policy optimization for multi-agent collaboration. *arXiv preprint arXiv:2603.21563v5*, 2026b.

Shuo Liu, Tianle Chen, Zeyu Liang, Xueguang Lyu, and Christopher Amato. LLM collaboration with multi-agent reinforcement learning. *arXiv preprint arXiv:2508.04652*, 2025.

Wenhua Nie, Jianan Wu, Junlin Liu, Ziwei Li, Zheng Lin, Zijian Zhang, et al. Gradient starvation in binary-reward GRPO: Why group-mean centering fails and why the simplest fix works. *arXiv preprint arXiv:2605.07689*, 2026.

Qwen Team. Qwen3.5: Towards native multimodal agents. https://qwen.ai/blog?id=qwen3.5, 2026.

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, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Gilbert Strang. On the construction and comparison of difference schemes. *SIAM journal on numerical analysis*, 5(3):506–517, 1968.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025. arXiv:2406.04692.

Xihuai Wang, Zheng Tian, Ziyu Wan, Ying Wen, Jun Wang, and Weinan Zhang. Order matters: Agent-by-agent policy optimization. In *International Conference on Learning Representations*, 2023a.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023b. arXiv:2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, pp. 24824–24837, 2022. arXiv:2201.11903.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling (COLM)*, 2024. arXiv:2308.08155.

Yi Xie, Siao Liu, Falong Fan, Yuanqi Yao, Siyang Cao, Yue Zhao, and Bo Liu. TeamTR: Trust-region fine-tuning for multi-agent LLM coordination. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026a.

Yi Xie, Yangyang Xu, Yi Fan, and Bo Liu. SAT: Sequential agent tuning for coordinator-free plug-and-play multi-LLM training with monotonic improvement guarantees. In *Proceedings of the 25th International Conference on Autonomous Agents and Multiagent Systems*, 2026b.

Zelai Xu, Zhexuan Xu, Ruize Zhang, Chunyang Zhu, Shi Yu, Weilin Liu, Quanlu Zhang, Wenbo Ding, Chao Yu, and Yu Wang. WideSeek-R1: Exploring width scaling for broad information seeking via multi-agent reinforcement learning. *arXiv preprint arXiv:2602.04634*, 2026.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*, 2023. arXiv:2305.10601.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025)*, 2025.

Yifan Zeng, Yiran Wu, Yaolun Zhang, Wentian Zhao, Kun Wan, Qingyun Wu, and Huazheng Wang. When does multi-agent rl improve llm workflows? workflow, scale, and policy-sharing tradeoffs. *arXiv preprint arXiv:2605.24202*, 2026.

Chenchen Zhang. Reinforcement learning for LLM-based multi-agent systems through orchestration traces. *arXiv preprint arXiv:2605.02801*, 2026.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62743–62767, 2024.

APPENDIX: AUTHOR STATEMENTS

AI USE STATEMENT

We used AI assistants, including Codex and GPT-based image generation, during the preparation of this manuscript. Their contributions included drafting and revising prose, clarifying theoretical exposition, checking notation and consistency, suggesting experimental designs, and assisting with analysis and interpretation of the experimental results. AI tools also supported figure design, schematic image generation, plotting-code development, and document layout. The authors directed these activities, selected the analyses and presentation, and took responsibility for the paper's claims, results, citations, and final content.

REPRODUCIBILITY STATEMENT

Appendices B and C prove the theoretical claims, and Appendix B.3 works the delegation example in full. Appendix C.1 specifies the implementation and the per-group logging protocol. Appendices D.1 to D.4 give the harness, policy ownership, data construction, the common SFT initialization, and the learner settings of Table 4. Appendix D defines the paired intervals and lists the run metadata, and Appendix D.5 gives the measurement and public evaluation protocols. Table 3 maps each figure to its run identifiers, evaluation units, and budgets. Figure 18 and Table 1 pair training seeds across controlled configurations. Figure 14 reports all declared condition–budget cells with a common comparison family; Figure 15 pairs starting teams and repeats across programs. The plotting package reproduces numerical summaries, tables, and data figures from the evaluation records.

ETHICS STATEMENT

The controlled studies use COMMAQA-N, SFT uses MuSiQue and Nemotron terminal sources alongside authored and verified execution demonstrations, and RL uses TMax-15K. Evaluation uses WebWalkerQA, Terminal-Bench 2.1, and BrowseComp-Plus. These are public task collections, and the work involves no human subjects and collects no personal data. Terminal execution runs through Harbor in containers under the official task tests, with task-specific resource limits, and the comparisons use fixed inference limits (Appendix D.5.3). Trained agent teams with tool access carry the general risks of autonomous agents; the paired audit is a training-time capability check and not a deployment safeguard.

A Extended Related Work **15**

B Theory and Proofs **15**

B.1 Notation and Quantitative Assumptions 15
B.2 Proof of Theorem 1 17
B.3 The Worked Example in Full 18
B.4 Proof of Theorem 2 19
B.5 Coverage Across Parent States 21
B.6 Smooth Coverage Costs and Update Order 22
B.7 Local Analysis of Ordered Update Programs 23
B.8 Supporting Lemmas for Ordered Programs 26
B.9 Proof of Theorem 5 27
B.10 Proofs of Update Corollaries 27
B.11 Proof of Proposition 3 28

C LACA Details **29**

C.1 Implementation Details 29
C.2 Coverage Estimation and Controller Guarantees 30
C.3 Proofs for Coverage Estimation 33
C.4 Proof of Theorem 6 33
C.5 Program Estimates and Calibration 34
C.6 Fixed-Budget and Compute Accounting 35

D Experimental Details **35**

D.1 Harness and Policy Ownership 35
D.2 Data Construction 37
D.3 Common SFT and Example Trajectories 37
D.4 Training Configuration 38
D.5 Measurements and Public Evaluation 40
D.6 Controlled Study Details 42

E Additional Results **42**

E.1 Controlled-Study Detail 42
E.2 Program Selection and Order Diagnostics 45
E.3 Controlled Continuation 47
E.4 Component Ablation 49
E.5 Downstream Training and Rollback 50
E.6 Case Studies of Routing and Execution 52

A Extended Related Work

Invocation depth and the number of trainable policies are distinct: shared-policy systems can create deep call graphs, whereas separate adapters or models introduce independently evolving states. The ownership map in Section 3 represents both settings.

Sequential multi-agent methods provide the closest update machinery. HAPPO and related trust-region methods control a sequence on a stage-start batch, while A2PO selects subsequent agents using corrected advantages on that batch (Kuba et al., 2022; Wang et al., 2023a). SAT and TeamTR provide certified or refreshed commits for agentic teams (Xie et al., 2026b;a). Safe policy improvement provides a related perspective (Laroche et al., 2019; Cho et al., 2025). LACA selects ordered policy subsets using team value and the coverage available to later owners. C3 uses restored histories to estimate counterfactual credit (Chen et al., 2026); LACA uses saved-parent entries to allocate usable updates and evaluates how each program changes subsequent learning. CCPO constructs role-specific credit (Li et al., 2026b), which affects the signal factor v_i. Coverage is measured under the chosen credit rule. OPSD transfers signals from privileged contexts within one model (Zhao et al., 2026). The commutator form of a two-map order error is classical in numerical analysis (Strang, 1968); our smooth decomposition uses it to quantify how one policy update changes another's effect.

GRPO's group baseline requires score variation for a learning signal (Shao et al., 2024); DAPO samples until enough prompts meet this condition (Yu et al., 2025). Delegation adds reach, completion, and ownership conditions at each policy, so collecting varied root outcomes need not supply a nested owner with usable updates.

Dynamic-structure methods add experts, tools, or agents. LACA separates proposal from admission: a proposed policy receives a fixed training and audit protocol (Appendix C.1.2), and admission depends on the resulting team capability within the remaining budget.

B Theory and Proofs

B.1 Notation and Quantitative Assumptions

General-task, multi-turn, and long-horizon capability are evaluated on separate distributions $\mathcal{D}_k$ with fixed inference limits:

$$Q_k(\mathsf{G}, \Theta) = \mathbb{E}_{x\sim\mathcal{D}_k}\mathbb{E}_{\tau\sim P_{\mathsf{G},\Theta}(\cdot|x)}\big[S_k(\tau)\big], \qquad k \in \{\mathrm{gen}, \mathrm{mt}, \mathrm{lh}\}, \tag{3}$$

where S_k is the task score under fixed maximum tokens, turns, tool calls, and sampling settings, and the body writes Q for the selected coordinate. Table 2 collects the notation.

Table 2: Notation used in the main text.

Symbol	Meaning
$\mathsf{G} = (\mathcal{N}, E), b, \mathcal{R}$	agent graph, agent-to-policy map, trainable policies
$\Theta, \mathcal{F}_t, B_t, D_t$	trainable state, system state, root budget, data law
$Q_{\mathrm{gen}}, Q_{\mathrm{mt}}, Q_{\mathrm{lh}}$	capability on evaluation tasks
$Q^{\mathrm{aud}}, Q^{\mathrm{test}}$	the selected coordinate on the audit and test splits
A, F, V, W, Z	activation, finish, nonzero advantage, usable tokens, effective group
$\rho_i, f_i, v_i, \nu_i, s_i$	reach, completion, signal availability, ownership, coverage
$s_c^{\mathrm{ev}}, Y_{n,c}, \mu_c$	audit exposure, evidence score, audit outcome
$\kappa, v(q)$	rollout group size and nonzero-advantage rate (worked example)
$\mathcal{A}, \sigma, \mathcal{P}_t, \mathcal{S}_t$	selected policies, order, update and audit action sets
$\Delta_i, \Sigma_i, \delta$	population field (distinct from a committed update Δ_j), noise covariance, trust radius
B_i, I_{ij}, Ω_{ij}	smooth direct, selection, and order values
$\Psi_{B,m}, \mathcal{G}_{B,m}$	valid-update probability and expected candidate gain
$\lambda_{j\to c}, \Lambda_{J\to c}$	single and joint coverage multipliers on policy c (λ_j, Λ_J for fixed c)
$\Gamma_t^{(2)}, \omega_t$	two-stage program value, reference continuation
$N_c^{\mathrm{id}}, B_{\mathrm{rem}}$	root-group evidence requirement and remaining root-group budget

B.1.1 QUANTITATIVE ASSUMPTIONS

Two-stage bookkeeping in Assumption 2. The direct gain g_j of an upstream update is counted over both stages under the reference continuation and excludes its effect through the coverage response of c. An upstream update acts on the coverage of c in every stage after its commit and, when it precedes the update of c in the program order, in the stage of its commit as well. A program is valued by Eq. (1): under the reference continuation ω_t, the declared update of c is applied in the stage of commitment and again in the stage that follows, with no further upstream commits. The STOP reference applies no update in either stage. Apart from the declared coverage response, the finite-batch model is additive. The smooth analysis in Appendix B.7 retains curvature, noise, and pair interactions. If each compared program's value changes by at most η under these additional terms, any strict comparison with margin greater than 2η keeps its sign. Thus robustness depends on the actual finite-B comparison margins, not just on the branch gap $G_c - \underline{G}_c$.

Evidence protocol in Assumption 1. For root task n, the audit pairs candidate and control rollout groups. The observation $Y_{n,c}$ is the difference between their mean final team rewards on the declared candidate-completion event (Appendix C.1.1). A singleton audit has one rollout per branch. The quantities B, N, and m_c count groups; physical rollouts are counted separately.

Smooth update model. Work on a Euclidean state space with orthogonal per-policy subspaces $E_1, \dots, E_L$. Each field Δ_i has support only on E_i. For all states in a radius-r neighborhood of one declared action, assume

$$\|\nabla Q\| \le M_1, \qquad \left\|\nabla^2 Q\right\| \le M_2, \qquad \left\|\nabla^3 Q\right\| \le M_3,$$

$$\|\Delta_i\| \le c_0\sqrt{\delta}, \qquad \|D\Delta_i\| \le c_1\sqrt{\delta},$$

and

$$\|D\Delta_i(z) - D\Delta_i(z')\| \le c_2\sqrt{\delta}\,\|z - z'\|\,.$$

The maximum declared update length $\ell_{\max}$ satisfies $\ell_{\max} c_0\sqrt{\delta} \le r$. The local stochastic analysis considers programs whose realized update segments remain in this neighborhood almost surely.

For the realized increment $\widehat{\Delta}_i(z) = \Delta_i(z) + \varepsilon_i$, assume

$$\mathbb{E}[\varepsilon_i \mid z] = 0, \qquad \mathbb{E}[\varepsilon_i \varepsilon_i^\top \mid z] = \delta\Sigma_i(z),$$

$$\|\Sigma_i(z)\| \le c_3, \qquad \|\Sigma_i(z) - \Sigma_i(z')\| \le c_4\,\|z - z'\|\,,$$

and

$$\mathbb{E}[\|\varepsilon_i\|^3 \mid z] \le c_5\delta^{3/2}.$$

Noises from distinct commits have zero conditional cross-covariance given the states at which they are drawn. These assumptions imply the uniform remainder bounds in the local analysis below.

The effective-group model uses separate assumptions. The active update branch is determined by whether the declared effective-group condition is met. Within one fixed-policy window, the finite-batch results assume independence across root groups, conditional on the starting state. The coverage factorization itself is a chain rule: neither the four events nor the routing edges need be independent. Shared activation or truncation within a group is part of their joint distribution.

Correlated root groups. For $M = \sum_{n=1}^{B} Z_n$, arbitrary dependence still gives $\mathbb{E}[M] = \sum_n \mathbb{E}[Z_n]$, but

$$\mathrm{Var}(M) = \sum_n \mathrm{Var}(Z_n) + 2\sum_{n<r} \mathrm{Cov}(Z_n, Z_r).$$

For example, if all groups share one Bernoulli event H and $Z_n = H$, then $\Pr(M \ge m) = \Pr(H = 1) = s$ for $1 \le m \le B$, instead of a binomial tail. Correlation can therefore prevent the transition from sharpening with B. The binomial, waiting-time, and concentration claims require their stated independent-group assumptions; the coverage identity does not. One rollout group contains continuations sampled from the same task entry and behavior checkpoint. Later windows retain separate group IDs; empirical uncertainty clusters their records by source task. Shared failures across tasks require larger independent blocks, not treating their continuations as new samples.

B.2 Proof of Theorem 1

Lemma 1 (Effective-count and evidence concentration). *Let $Z_1, \dots, Z_N$ be independent Bernoulli variables of mean s and let $M_N = \sum_n Z_n$. Then*

$$\Pr(M_N < Ns/2) \le \exp(-Ns/8).$$

If, conditional on $Z_n = 1$, the variables $Y_n - \mu$ are independent σ^2-sub-Gaussian, then on $M_N \ge Ns/2$,

$$\Pr\left(\left|\frac{1}{M_N}\sum_n Z_n Y_n - \mu\right| > r \,\middle|\, M_N\right) \le 2\exp\left(-\frac{M_N r^2}{2\sigma^2}\right).$$

Proof. The first statement is the multiplicative Chernoff bound with relative deviation one half. Conditional on the indicators and hence on M_N, the estimator is the mean of M_N independent σ^2-sub-Gaussian variables. The two-sided sub-Gaussian bound proves the second claim. □

The quantitative form of the evidence bound is as follows. If $Ns_c^{\mathrm{ev}} \ge 8\log(4/\alpha)$, then with probability at least $1-\alpha$,

$$|\widehat{\mu}_{c,N} - \mu_c| \le 2\sigma_c\sqrt{\frac{\log(4/\alpha)}{Ns_c^{\mathrm{ev}}}}. \tag{4}$$

Consequently, the sufficient root-group requirement for error at most ϵ scales as $1/s_c^{\mathrm{ev}}$ (Eq. (5) in Appendix B.2). A matching lower bound (Appendix B.2.1) shows that the inverse-coverage dependence of training and the inverse-exposure dependence of identification are unavoidable up to constants.

Proof. The waiting time to the mth success in independent Bernoulli trials of probability s_c has a negative-binomial distribution with mean m/s_c. This proves the training-burden statement.

For a fixed number N of root groups, Lemma 1 applied to the exposure indicators gives

$$\Pr(M_{c,N} < Ns_c^{\mathrm{ev}}/2) \le \exp(-Ns_c^{\mathrm{ev}}/8).$$

When $Ns_c^{\mathrm{ev}} \ge 8\log(4/\alpha)$, this probability is at most $\alpha/4$. On the event $M_{c,N} \ge Ns_c^{\mathrm{ev}}/2$, set

$$r = 2\sigma_c\sqrt{\frac{\log(4/\alpha)}{Ns_c^{\mathrm{ev}}}}.$$

The conditional tail in Lemma 1 is then at most $2\exp(-\log(4/\alpha)) = \alpha/2$. A union bound proves Eq. (4) with probability at least $1-3\alpha/4$, which is at least $1-\alpha$. The sufficient root-group requirement for error at most ϵ is

$$N \ge \max\left\{\frac{8\log(4/\alpha)}{s_c^{\mathrm{ev}}}, \frac{4\sigma_c^2\log(4/\alpha)}{s_c^{\mathrm{ev}}\epsilon^2}\right\}; \tag{5}$$

The two requirements respectively ensure the count event and $r \le \epsilon$.

□

B.2.1 Matching lower bound

Proof of the lower bound. Consider two evidence worlds. In both worlds, the exposure indicator is Bernoulli(s_c^{ev}). Conditional on exposure, let $Y \sim N(+2\epsilon, \sigma_c^2)$ in the first world and $Y \sim N(-2\epsilon, \sigma_c^2)$ in the second; without exposure, the observation is identical and carries no evidence. The Kullback-Leibler divergence for one root group is

$$s_c^{\mathrm{ev}}\, D_{\mathrm{KL}}\left(N(+2\epsilon, \sigma_c^2) \,\|\, N(-2\epsilon, \sigma_c^2)\right) = \frac{8s_c^{\mathrm{ev}}\epsilon^2}{\sigma_c^2}.$$

For N independent root groups, the divergence is $8Ns_c^{\mathrm{ev}}\epsilon^2/\sigma_c^2$. The Bretagnolle-Huber inequality implies that the sum of the two sign errors is at least

$$\frac{1}{2}\exp\left(-\frac{8Ns_c^{\mathrm{ev}}\epsilon^2}{\sigma_c^2}\right).$$

An estimate with error at most ϵ has the correct sign in both worlds. Thus an estimator attaining this accuracy with probability $1-\alpha$ in each world gives a sign test with error at most α; the sum of errors is at most 2α. For $0 < \alpha < 1/4$, rearranging gives

$$N \geq \frac{\sigma_c^2}{8 s_c^{\mathrm{ev}} \epsilon^2} \log \frac{1}{4\alpha}.$$

This proves that inverse exposure is necessary up to constants. Under controlled activation, Assumption 1 changes only the activation factor of both rates, so replacing the passive path product by e_c gives the final claim. □

B.2.2 Proof of Corollary 1

Proof of Corollary 1. The group-level reach events along the declared path are nested. Applying the chain rule to their conditional probabilities gives

$$\rho_c^{\mathrm{pass}} = \prod_{h=1}^{d} p_h.$$

Multiplying by the conditional finish, nonzero-advantage, and usable-token probabilities gives the path-product form of s_c^{pass}, and multiplying by the finish probability alone gives the audit exposure. If every $p_h \leq \bar{p} < 1$, then $s_c^{\mathrm{pass}} \leq \bar{p}^d f_c v_c \nu_c$ and $s_c^{\mathrm{ev,pass}} \leq \bar{p}^d f_c$. Both the exact expected training requirement m/s_c and the upper and lower identification requirements of Theorem 1 are therefore bounded below by a constant times $\bar{p}^{-d}$. Substituting the upper bound on exposure into the lower-bound requirement and comparing it with B_{rem} yields the budget-dependent critical depth. A controlled intervention at the declared parent replaces the upstream path event by exposure probability e_c, so the product $\prod_h p_h$ disappears. □

B.3 The Worked Example in Full

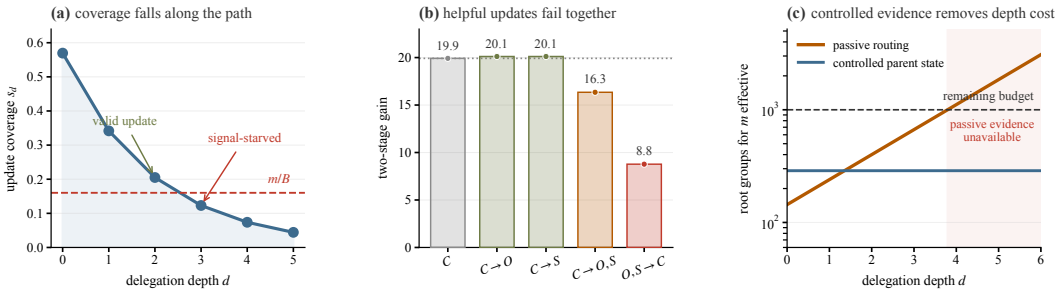

Figure 7: Update coverage in the worked example. **(a)** Coverage falls with delegation depth and crosses the finite-batch update boundary. **(b)** Either upstream update is helpful alone, while their joint coverage shift makes update-all worse than training the nested policy alone; bars and dots give exact calculated values. **(c)** Passive evidence cost grows with the inverse path product; controlled parent-state activation removes this depth factor.

Figure 7 expands the coverage and value calculations for the worked example in Section 4.2.

B.3.1 Program Value and Worked Example

In the chain of Section 4.2, root agent O reaches subagent S with probability p_1, after which S reaches nested subagent C with probability p_2. An activated prompt gives C a group of κ independent binary-reward rollouts with per-rollout success probability q. The group has nonzero relative advantage only when its rewards are mixed, with

$$v(q) = 1 - q^\kappa - (1-q)^\kappa, \tag{6}$$

so, when the completion and ownership factors of Section 3 equal one, C has update coverage $s = p_1 p_2 v(q)$. We use the two-stage program value of Eq. (1).

Equation (6) excludes the all-success and all-failure outcomes. For B root groups, the effective count is $M_C \sim \text{Binomial}(B, s)$. The update is valid when $M_C \geq m$, with probability $\Psi_{B,m}(s) = \Pr\{M_C \geq m\}$ and expected gain $\mathcal{G}(s) = \Delta\Psi_{B,m}(s)$. This instantiates Assumption 2 with $\Delta = G_c$ and $\underline{G}_c = 0$.

The two-round accounting works as follows. An update committed in round one changes the routing probabilities seen in round two, while the nested subagent's round-one coverage depends only on the updates committed before it within round one. Updating only C therefore earns $\mathcal{G}(s)$ in both rounds. Adding upstream update U_j in the order $C \to U_j$ leaves round one at coverage s and moves round two to $\lambda_j s$, for a total of $\mathcal{G}(s) + g_j + \mathcal{G}(\lambda_j s)$; any other order also degrades round one and is weakly worse, so C first is the most favorable order. The same argument gives $\mathcal{G}(s) + g_O + g_S + \mathcal{G}(\lambda_O \lambda_S s)$ for update-all with C first, and $g_O + g_S + 2\,\mathcal{G}(\lambda_O \lambda_S s)$ for the top-down order $U_O \to U_S \to C$, whose gap to the best order is exactly $\mathcal{G}(s) - \mathcal{G}(\lambda_O \lambda_S s)$. Subtracting the candidate-only value shows that update-all with C first is strictly worse than updating C alone precisely when the second inequality of Eq. (2) holds, while each single addition helps precisely when the first inequality holds.

With $\kappa = 8$ and $q = 0.1$, $v(0.1) = 1 - 0.1^8 - 0.9^8 \approx 0.5695$. The declared parameters $p_1 = p_2 = 0.6$ and $\lambda_O = \lambda_S = 0.85$ give coverage levels $s \approx 0.2050$, $\lambda s \approx 0.1743$, and $\lambda^2 s \approx 0.1481$, against the buffer fraction $m/B = 82/512 \approx 0.1602$. The exact binomial tails are $\Psi_{512,82}(0.2050) \approx 0.996$, $\Psi_{512,82}(0.1743) \approx 0.815$, and $\Psi_{512,82}(0.1481) \approx 0.239$. With $\Delta = 10$ and $g_O = g_S = 2$, the two-round gains are 19.92 for C only, 20.11 for either single addition, 16.34 for update-all with C first, and 8.77 top-down. Both conditions of Eq. (2) hold: $2 > 1.80$ and $4 < 7.57$.

The waiting time $\mathbb{E}[T_m] = m/s_d$ from Appendix B.3.1 is the mean of a negative binomial variable, the number of root groups until the m-th effective one, and instantiates Theorem 1 at $s_d = v(q)\prod_h p_h$, that is, with $v_c = v(q)$ and $f_c = \nu_c = 1$; the identification requirement of Eq. (5) carries the factor $1/s_d^{\text{ev}}$ with $s_d^{\text{ev}} = \prod_h p_h$, the same path product without the advantage factor. For the figure, $\bar{p} = 0.6$, $e = 0.5$, and a remaining budget of 1000 root groups give a passive demand of $m/(v(q)\bar{p}^{\,d}) \approx 144\,\bar{p}^{-d}$, a controlled demand of $m/(e\,v(q)) \approx 288$, and a critical depth $d_{\text{crit}} \approx 3.8$.

Finish and usable-token rates below one multiply coverage by $f_c\nu_c$. Non-binary verifier rewards change the signal-availability factor; the coverage product and the inequalities in Eq. (2) retain their form.

Remark 1 (Finite routed batches). For fixed B, the valid-update probability $\Psi_{B,m}(s)$ is a smooth polynomial in s. When $m_B/B \to \tau$, the law of large numbers makes it converge to one for $s > \tau$ and zero for $s < \tau$. Thus the cliff becomes sharp as the number of root groups grows while the required effective fraction stays fixed.

B.4 Proof of Theorem 2

The next remark records the large-batch regime in which the two conditions of Eq. (2) hold, establishing a nonempty failure region for Theorem 2.

Remark 2 (Large-batch existence). Let $|J| \geq 2$. Set $m = m_B$ with $m_B/B \to \tau \in (0, 1)$ and suppose

$$s_c\lambda_j > \tau \quad \text{for every } j \in J, \qquad s_c\Lambda_J < \tau. \tag{7}$$

For any positive gains g_j satisfying $\sum_{j\in J} g_j < G_c - \underline{G}_c$, the two inequalities in Eq. (2) hold for all sufficiently large B. Thus each separate addition improves the same candidate-only baseline, while their joint addition lowers team value. Under Eq. (7), the proof below establishes these limiting gain comparisons.

Proof. Under Assumption 2, the candidate-only program updates the candidate at coverage s_c in both stages and has expected capability gain $2\mathcal{G}_{B,m}(s_c)$. Adding only upstream update j in the candidate-first order leaves the first stage at s_c and moves the second to $s_c\lambda_j$, giving

$$\mathcal{G}_{B,m}(s_c) + g_j + \mathcal{G}_{B,m}(s_c\lambda_j).$$

The first line of Eq. (2) says this is strictly larger than the candidate-only gain for every j. Thus every one-at-a-time addition in the candidate-first order is beneficial. Committing all updates in J in the candidate-first order gives

$$\mathcal{G}_{B,m}(s_c) + \sum_{j\in J} g_j + \mathcal{G}_{B,m}(s_c\Lambda_J).$$

The second line of Eq. (2) says this is strictly smaller than $2\mathcal{G}_{B,m}(s_c)$. Any other order replaces the first-stage term $\mathcal{G}_{B,m}(s_c)$ by $\mathcal{G}_{B,m}$ evaluated at the coverage $s_c\Lambda_{J'}$ of the prefix set $J' \subseteq J$ committed before c, which is no larger than s_c because $\Lambda_{J'} \leq \Lambda_\emptyset = 1$, while the second-stage term stays at $s_c\Lambda_J$ because the coverage depends only on the set of commits in effect, and $\mathcal{G}_{B,m}$ is nondecreasing in its argument because $\Psi_{B,m}$ is and $G_c > \underline{G}_c$, so candidate-first is the most favorable order and update-all loses in every order. This proves the finite-B claim.

For the asymptotic existence claim, let $m_B/B \to \tau$. For any fixed $x > \tau$, the weak law of large numbers gives

$$\Psi_{B,m_B}(x) = \Pr\left(\frac{1}{B}\,\mathrm{Binomial}(B,x) \geq \frac{m_B}{B}\right) \longrightarrow 1.$$

For fixed $x < \tau$, the same probability tends to zero. The conditions in Eq. (7) therefore imply

$$\mathcal{G}_{B,m_B}(s_c) \to G_c, \qquad \mathcal{G}_{B,m_B}(s_c\lambda_j) \to G_c$$

for every j, while

$$\mathcal{G}_{B,m_B}(s_c\Lambda_J) \to \underline{G}_c.$$

Hence each one-at-a-time marginal tends to $g_j > 0$, whereas the update-all difference from candidate-only tends to

$$\sum_{j\in J} g_j + \underline{G}_c - G_c < 0.$$

Both signs therefore hold for all sufficiently large B. □

Proposition 1 (A finite-budget failure region)**.** *Under Assumption 2, let J be a finite set with $|J| \geq 2$ and $\Lambda_\varnothing = 1$. Set $\tau = m_c/B$, where $1 \leq m_c \leq B$, and $\Delta_G = G_c - \underline{G}_c > 0$. Write $\Gamma(u) = \Gamma_t^{(2)}(u;\omega_t)$, let u_c update only c, let $u_j = (c,j)$, and let $u_{J,\sigma}$ update $\{c\} \cup J$ in order σ. Suppose that, for some $\gamma > 0$,*

$$s_c\lambda_j \geq \tau + \gamma \quad \textit{for every } j \in J, \qquad s_c\Lambda_J \leq \tau - \gamma.$$

With $\epsilon_B = \exp(-2B\gamma^2)$,

$$\Gamma(u_j) - \Gamma(u_c) \geq g_j - \Delta_G\epsilon_B,$$
$$\max_\sigma \Gamma(u_{J,\sigma}) - \Gamma(u_c) \leq \sum_{j\in J} g_j - \Delta_G(1-2\epsilon_B).$$

Thus, if $g_j > \Delta_G\epsilon_B$ for every j and $\sum_j g_j < \Delta_G(1-2\epsilon_B)$, each separate addition improves team value, while their joint addition lowers it in every order.

Proof. Write $\Psi(p) = \Pr\{\mathrm{Binomial}(B,p) \geq m_c\}$. Hoeffding's inequality gives $1 - \Psi(p) \leq \epsilon_B$ for $p \geq \tau + \gamma$ and $\Psi(p) \leq \epsilon_B$ for $p \leq \tau - \gamma$. Since J is nonempty and $\lambda_j \leq 1$, the singleton premise also gives $s_c \geq \tau + \gamma$. Hence

$$\mathcal{G}_{B,m_c}(s_c) - \mathcal{G}_{B,m_c}(s_c\lambda_j) \leq \Delta_G\epsilon_B,$$
$$\mathcal{G}_{B,m_c}(s_c) - \mathcal{G}_{B,m_c}(s_c\Lambda_J) \geq \Delta_G(1-2\epsilon_B).$$

The first bound follows by substituting the first inequality into the singleton value difference in the proof above. The second follows by substituting the second inequality into the joint value difference with c first, which attains the maximum over orders under Assumption 2. □

For two upstream commits, take $g_1 = g_2 = \Delta_G/4$ and $\epsilon_B \leq 1/8$. Each singleton then gains at least $\Delta_G/8$, while the best joint order loses at least $\Delta_G/4$. The coverage premises can hold simultaneously: $B = 512$, $m_c = 256$, $s_c = 0.8$, $\lambda_1 = \lambda_2 = 0.75$, $\Lambda_{\{1,2\}} = 0.5625$, and $\gamma = 0.05$ give singleton coverage 0.6, joint coverage 0.45, and $\epsilon_B = e^{-2.56} < 1/8$. The earlier worked example is certified by the exact binomial criterion in Eq. (2); these exponential bounds are sufficient.

B.4.1 Proof of Corollary 2

Proof. The first update of c sees coverage $s_c\Lambda_{J_{\text{pre}}}$, and its second update sees $s_c\Lambda_J$. Subtracting the value of u_c under the same continuation gives

$$\begin{aligned}\Gamma(u_{J,\sigma}) - \Gamma(u_c) &= \sum_{j\in J} g_j - 2\mathcal{G}_{B,m_c}(s_c) \\ &\quad + \mathcal{G}_{B,m_c}(s_c\Lambda_{J_{\text{pre}}}) + \mathcal{G}_{B,m_c}(s_c\Lambda_J) \\ &= \sum_{j\in J} g_j - C(J) - C(J_{\text{pre}}).\end{aligned}$$

Monotone coverage and the monotonicity of $\mathcal{G}_{B,m_c}$ imply $C(J_{\text{pre}}) \geq 0$. Placing c first gives $J_{\text{pre}} = \varnothing$ and $C(\varnothing) = 0$, so $\max_\sigma \Gamma(u_{J,\sigma}) - \Gamma(u_c) = \sum_j g_j - C(J)$. □

For a single upstream commit j, the difference between the candidate-first and upstream-first orders is $C(\{j\}) = \mathcal{G}_{B,m_c}(s_c) - \mathcal{G}_{B,m_c}(s_c\lambda_j)$. If $s_c > \tau > s_c\lambda_j$ and $m_c = m_B$ with $m_B/B \to \tau$, the weak law of large numbers gives $\Psi_{B,m_B}(s_c) \to 1$ and $\Psi_{B,m_B}(s_c\lambda_j) \to 0$. The order gap therefore tends to $G_c - \underline{G}_c$: order can change whether a downstream policy receives a valid update even when the smooth pair interaction is small.

B.5 Coverage Across Parent States

Repeated sampling from a saved parent state separates the number of independent contexts from the number of groups collected at each context. Consider K independent contexts, each comprising a task, parent state, and environment. Context k has usable-group probability P_k and supplies r root groups, so $B = Kr$. The within-group rollout count remains κ. In the following model,

$$P_k \overset{\text{iid}}{\sim} \text{Beta}(a,b), \qquad Z_{k\ell} \mid P_1,\dots,P_K \sim \text{Bernoulli}(P_k), \qquad a,b>0, \tag{8}$$

with conditional independence across k,ℓ. Writing $M_k = \sum_{\ell=1}^r Z_{k\ell}$ and $M = \sum_k M_k$, each M_k is beta-binomial; M is the sum of these independent counts. Thus

$$s = \frac{a}{a+b}, \qquad \chi = \frac{1}{a+b+1}, \qquad \mathbb{E}M = Bs, \qquad \text{Var}(M) = Bs(1-s)\{1+(r-1)\chi\}. \tag{9}$$

Here χ is the correlation between two indicators sharing a context. Resampling an independent context for every root group instead gives independent Bernoulli indicators of mean s. For fixed K and $r \to \infty$, the block model gives $M/B \to \bar{P}_K = K^{-1}\sum_k P_k$ almost surely. Consequently, if $m_B/B \to \tau \in (0,1)$, then $\Pr(M \geq m_B) \to \Pr(\bar{P}_K \geq \tau)$: repeating groups within a fixed number of contexts retains variation between sampling windows.

From usable counts to an update. Let the downstream policy's local objective be $Q_c(x) = -h(x-x^*)^2/2$, with $h>0$ and $0<\eta h<1$. Conditional on the current state, context probabilities, and $M = n > 0$, assume the sampled update direction has mean $Q_c'(x)$ and variance σ^2/n, where $\sigma^2 > 0$. The learner takes $x^+ = x + \eta\widehat{g}_n$ for $n>0$ and keeps x when $M=0$. For two updates below, counts and noise are freshly sampled at each stage, and the coverage law is independent of x. These assumptions describe an average over usable contributions. GRPO and DAPO instead use the token-based reduction in Appendix D.4: zero-advantage owner tokens remain in the denominator. Their conditional mean and variance therefore depend on loss weights and denominators as well as on M.

Proposition 2 (Local gain from usable groups)**.** *Set $A_c = \eta(1-h\eta/2)[Q_c'(x)]^2$, $C_c = h\eta^2\sigma^2/2$, $p_B = \Pr(M=0)$, and $u_B = \mathbb{E}[\mathbf{1}\{M>0\}/M]$. Under the local update model,*

$$\mathcal{G}_B^c := \mathbb{E}[Q_c(x^+) - Q_c(x)] = A_c(1-p_B) - C_c u_B, \qquad \mathcal{R}_B^c := A_c - \mathcal{G}_B^c = A_c p_B + C_c u_B. \tag{10}$$

Proof. The exact quadratic expansion is $Q_c(x+\eta\widehat{g}) - Q_c(x) = \eta Q_c'(x)\widehat{g} - h\eta^2\widehat{g}^2/2$. Its conditional expectation is $A_c - C_c/n$ for $M=n>0$ and zero for $M=0$. Averaging gives the result. □

Lemma 2 (Reciprocal usable count)**.** *For a sum M of independent Bernoulli variables with mean $\mu>0$, there are universal constants $c_0, C_0 > 0$ such that*

$$c_0 \min\{\mu, \mu^{-1}\} \leq \mathbb{E}[\mathbf{1}\{M>0\}/M] \leq C_0 \min\{\mu,\mu^{-1}\}. \tag{11}$$

Proof. Cauchy–Schwarz gives $\mu\mathbb{E}[\mathbf{1}\{M > 0\}/M] \geq \Pr(M > 0)^2$, and independence gives $\Pr(M = 0) \leq e^{-\mu}$. These imply the lower bound. For $\mu \leq 1$, the expectation is at most $\Pr(M > 0) \leq \mu$. For $\mu \geq 1$, splitting at $\mu/2$ and applying the multiplicative Chernoff bound gives the upper bound $2/\mu + \Pr(M < \mu/2) \leq 2/\mu + e^{-\mu/8} \leq C_0/\mu$. □

Theorem 3 (Learning cost and the low-coverage tail)**.** *Under (8) and the local update model, fix K, a, b and a state with $Q_c'(x) \neq 0$. As $B = Kr \to \infty$,*

$$\mathcal{R}_B^c = \begin{cases} \Theta(B^{-Ka}), & Ka < 1, \\ \Theta((\log B)/B), & Ka = 1, \\ \Theta(B^{-1}), & Ka > 1. \end{cases} \tag{12}$$

Proof. The probability of obtaining no usable group is

$$p_B = \left[\frac{\mathrm{B}(a, b+r)}{\mathrm{B}(a,b)}\right]^K \sim \left[\frac{\Gamma(a+b)}{\Gamma(b)}\right]^K r^{-Ka}.$$

The beta density near zero is proportional to p^{a-1}. Convolution and rescaling give $f_{\bar{P}_K}(p) \sim D_K p^{Ka-1}$, where

$$D_K = \frac{K^{Ka}}{\Gamma(Ka)}\left[\frac{\Gamma(a+b)}{\Gamma(b)}\right]^K > 0.$$

Conditional on the contexts, M is a sum of independent Bernoulli variables with mean $B\bar{P}_K$. Lemma 2 therefore bounds u_B, up to constant factors, by the integral of $\min\{Bp, (Bp)^{-1}\}$ against this density. For small fixed $p_* > 0$, the contribution near zero has order

$$B\int_0^{1/B} p^{Ka}\,dp + \frac{1}{B}\int_{1/B}^{p_*} p^{Ka-2}\,dp.$$

The remainder is $O(B^{-1})$ and includes positive probability mass giving an $\Omega(B^{-1})$ contribution. Evaluating the integrals and using $A_c, C_c > 0$ proves (12). □

The result describes the expected gain deficit of one local update. The mean coverage s does not determine its rate: the lower tail and the number of independent parent contexts also matter. With the same context population, a larger fixed K can move Ka above one and recover the inverse-budget rate. If instead $\Pr(P_k = 0) = \zeta > 0$, then $\Pr(M = 0) \geq \zeta^K$ and $\mathcal{R}_B^c \geq A_c\zeta^K$ for a fixed context count.

B.6 Smooth Coverage Costs and Update Order

The same local model yields joint update costs without an imposed effective-count threshold. Suppose upstream commit j contributes direct final gain $g_j > 0$ and otherwise changes the downstream policy only by thinning every context probability to $\lambda_j P_k$, with $0 < \lambda_j < 1$. A set J has multiplier $\Lambda_J = \prod_{j\in J}\lambda_j$. The downstream policy updates twice: the first update sees preceding commits, and the second sees all commits. Both stages draw fresh independent context pools. Refreshing continuations at a saved entry conditions on that entry; independent context pools additionally require independent sampling of parent states between stages. Thinning uses the same underlying beta law; λP_k has its support on $[0, \lambda]$.

For the count M_λ under multiplier λ, write $p_\lambda = \Pr(M_\lambda = 0)$, $u_\lambda = \mathbb{E}[\mathbf{1}\{M_\lambda > 0\}/M_\lambda]$, $\gamma = 1 - \eta h$, and $c_\lambda = \gamma^2 + (1-\gamma^2)p_\lambda$. Starting at error $e_0 = x_0 - x^*$, the two-update gain is exactly

$$V(\lambda_1, \lambda_2) = \frac{h}{2}\left[e_0^2(1 - c_{\lambda_1}c_{\lambda_2}) - \eta^2\sigma^2(c_{\lambda_2}u_{\lambda_1} + u_{\lambda_2})\right]. \tag{13}$$

Indeed, one update gives $\mathbb{E}e_1^2 = c_{\lambda_1}e_0^2 + \eta^2\sigma^2 u_{\lambda_1}$; applying the same recurrence at the fresh second stage proves the formula.

Theorem 4 (Superadditive downstream learning cost)**.** *Under the preceding thinning and local update model, suppose $Ka > 1$ and all multipliers are fixed. Set $H_K = \mathbb{E}[\bar{P}_K^{-1}] < \infty$ and $D_J = V(1,1) - V(1, \Lambda_J)$. For fixed K and $B = Kr \to \infty$,*

$$D_J = \frac{C_c H_K}{B}(\Lambda_J^{-1} - 1) + o(B^{-1}), \tag{14}$$

$$D_{\{1,2\}} - D_{\{1\}} - D_{\{2\}} = \frac{C_c H_K}{B}(\lambda_1^{-1} - 1)(\lambda_2^{-1} - 1) + o(B^{-1}) > 0 \tag{15}$$

for sufficiently large B. There is consequently a nonempty region of positive direct gains for which each upstream addition is helpful alone and their combination is worse than the downstream updates alone.

Proof. Conditional independence gives $p_\lambda \le \mathbb{E}e^{-\lambda B \bar{P}_K} = O(B^{-Ka}) = o(B^{-1})$, using the density near zero from Theorem 3. Conditional concentration and Lemma 2 give

$$B\mathbb{E}[\mathbf{1}\{M_\lambda > 0\}/M_\lambda \mid P_1, \ldots, P_K] \longrightarrow (\lambda \bar{P}_K)^{-1}.$$

The same lemma bounds the left side by a constant times $(\lambda \bar{P}_K)^{-1}$, which is integrable because $Ka > 1$. Dominated convergence gives $u_\lambda = H_K/(\lambda B) + o(B^{-1})$, while $c_\lambda = \gamma^2 + o(B^{-1})$. Substitution into (13) proves (14); expanding $(\lambda_1 \lambda_2)^{-1}$ proves (15). Let $\Delta_B = D_{\{1,2\}} - D_{\{1\}} - D_{\{2\}} > 0$ and choose $g_j = D_{\{j\}} + \Delta_B/4$. Each singleton improves the program by $\Delta_B/4$, while their combination reduces it by $\Delta_B/2$. □

Order and scale. If $\lambda_{\text{pre}} < 1$ is the multiplier of commits preceding the downstream policy and Λ is the final multiplier, the same substitution gives

$$V(1, \Lambda) - V(\lambda_{\text{pre}}, \Lambda) = \frac{\gamma^2 C_c H_K}{B}(\lambda_{\text{pre}}^{-1} - 1) + o(B^{-1}) > 0. \tag{16}$$

Updating the downstream policy first is therefore best among these coverage-decreasing orders for sufficiently large B. The loss is of order B^{-1}, and fixed positive direct gains eventually exceed it; the constructive gain region above can shrink with B. An upstream update that increases coverage or changes useful gradient directions or partner responses can favor a different order.

A finite-budget example. Take $K = 1$, $B = 512$, $P \sim \text{Beta}(4, 15.5091687455)$, $\lambda_1 = \lambda_2 = 0.85$, $h = 1$, $\eta = 0.5$, $\sigma^2 = 100$, $e_0 = 1$, and $g_1 = g_2 = 0.0289$. Integrating the conditional binomial probabilities against this beta density gives the following gains:

Downstream only:	0.27742905,	Then one upstream commit:	0.27871925,
Then both upstream commits:	0.27489469,	Both upstream commits first:	0.25981083.

Here $D_{\{1\}} = D_{\{2\}} = 0.02760979$ and $D_{\{1,2\}} = 0.06033435$. Each direct gain exceeds its individual cost, while their sum is below the joint cost. These values are gains in the specified quadratic model, with no effective-count threshold. A comparison that averages over all B groups instead uses $\widehat{g}^{\text{all}} = (M/B)\widehat{g}_M$. Keeping the same parameters and noise assumptions gives $D_{\{1,2\}} - D_{\{1\}} - D_{\{2\}} = -0.001380$. This model comparison shows that superadditivity depends on the normalization.

B.7 Local Analysis of Ordered Update Programs

This appendix develops a local value model for ordered update programs on one fixed active update branch. The model separates direct gains, curvature, noise, and interactions between roles, providing a local account of program value alongside the finite-batch coverage analysis. A certified macro-update of policy i has realized increment $\widehat{\Delta}_i = \Delta_i + \varepsilon_i$, where the population field $\Delta_i(\Theta) = \mathbb{E}_{\xi \sim P_{G,\Theta}}[h_i(\xi, \Theta)]$ has scale $\|\Delta_i\| = O(\sqrt{\delta})$ and ξ contains the rollout group, teacher targets, advantages, clipping masks, and update statistics. For any realized update whose line segment stays in this neighborhood, L-smoothness gives $Q(\Theta + \widehat{\Delta}_i) - Q(\Theta) \ge \langle \nabla Q(\Theta), \widehat{\Delta}_i \rangle - L \left\| \widehat{\Delta}_i \right\|^2 /2$. Its conditional expectation links update direction to team gain, complementing the coverage analysis in Section 4.2. The expansion below resolves its direction, curvature, and noise terms across ordered updates.

Assumption 3 (Regular local updates)**.** Every realized update segment remains in the declared neighborhood almost surely. On this neighborhood, Q is three times continuously differentiable, every active population field is differentiable with a Lipschitz derivative, the field and derivative scales are $O(\sqrt{\delta})$, and update noise has zero conditional mean, conditional covariance $\delta \Sigma_i(\Theta)$, bounded third moment, and zero conditional cross-covariance across commits. The quantitative bounds are stated in Appendix B.1.1.

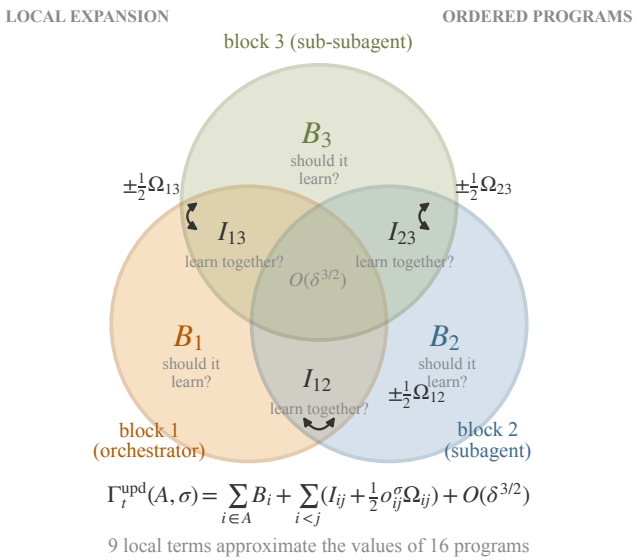

Figure 8: The smooth value of an ordered program as a three-set diagram. Exclusive regions carry the direct values B_i, pairwise overlaps carry the symmetric selection values I_{ij} and the sign-changing order values $\pm\frac{1}{2}\Omega_{ij}$, and the triple overlap enters the $O(\delta^{3/2})$ remainder, so nine quantities value all sixteen ordered programs of a three-policy system (Theorem 5).

Assumption 4 (Matched rollout and evaluation budgets)**.** Every compared program receives the same per-update root rollout budget, task identities, sampling settings, and test evaluation budget. A selected policy applies the same update rule whenever it receives the same effective data, and order comparisons use matched seeds and paired evaluation. The value comparison does not convert optimizer steps, tokens, or wall time into an abstract value cost.

One number per policy and two numbers per pair describe the smooth part of every ordered program. Let $g = \nabla Q(\Theta)$ and $H_Q = \nabla^2 Q(\Theta)$ at the stage-start state. For policy i, the direct capability value is

$$B_i = \langle g, \Delta_i \rangle + \tfrac{1}{2}\Delta_i^\top H_Q \Delta_i + \tfrac{1}{2}\delta \operatorname{tr}(H_Q \Sigma_i), \tag{17}$$

where the last term is the leading expected effect of update noise. For policies i and j, the symmetric selection value is

$$I_{ij} = \Delta_i^\top H_Q \Delta_j + \tfrac{1}{2}\left\langle g, D\Delta_j[\Delta_i] + D\Delta_i[\Delta_j] \right\rangle, \tag{18}$$

and the antisymmetric order value is

$$\Omega_{ij} = \left\langle g, D\Delta_j[\Delta_i] - D\Delta_i[\Delta_j] \right\rangle. \tag{19}$$

These quantities describe whether one policy improves the selected capability, whether two smooth updates are complementary, and which update should see the other policy's fresh data first (Figure 8).

Let $\Gamma_t^{\mathrm{upd}}(\mathcal{A}, \sigma)$ denote the expected change in Q produced by the current ordered program, before the next-stage continuation.

Theorem 5 (Ordered update program decomposition)**.** *Under Assumptions 3 and 4, let $\sigma = (i_1, \ldots, i_\ell)$ be an ordered program with selected set $\mathcal{A}$, and let $o_{ij}^{\sigma} = 1$ when i appears before j and $o_{ij}^{\sigma} = -1$ otherwise. There is a constant K_{upd}, independent of δ and the program, such that on one fixed active update branch,*

$$\begin{aligned} \Gamma_t^{\mathrm{upd}}(\mathcal{A}, \sigma) &= \sum_{i\in\mathcal{A}} B_i + \sum_{\substack{i<j \\ i,j\in\mathcal{A}}} \left(I_{ij} + \tfrac{1}{2}o_{ij}^{\sigma}\Omega_{ij}\right) + R_{\mathcal{A},\sigma}, \\ |R_{\mathcal{A},\sigma}| &\le K_{\mathrm{upd}}\ell^3\delta^{3/2}. \end{aligned} \tag{20}$$

For a fixed pair, the symmetric half survives a swap while the antisymmetric half changes sign. The complete path and noise argument is in Appendix B.9.

Corollary 3 (Insertion value). *Let $u = (\mathcal{A}, \sigma)$ be a program and let $u \oplus_p i$ insert an unselected policy i at position p. Its leading marginal capability value is*

$$M_{i|\mathcal{A},p} = B_i + \sum_{j \in \mathcal{A}} \left(I_{ij} + \tfrac{1}{2} o_{ij}^{p} \Omega_{ij} \right). \tag{21}$$

If $\max_p M_{i|\mathcal{A},p}$ is below the negative remainder bound, no insertion of i improves the smooth program; if $\min_p M_{i|\mathcal{A},p}$ is above the bound, every insertion improves it. Near the coverage boundary, this smooth test must be combined with the effective-group condition of Theorem 2.

Remark 3 (Scale of the three decisions). Under Assumption 3, B_i is usually $O(\sqrt{\delta})$ while I_{ij} and Ω_{ij} are $O(\delta)$ on a fixed active branch. Theorem 2 identifies a different effect: crossing the finite-batch effective-group threshold changes which branch is active and can create an $O(1)$ capability difference even when all smooth singleton gains are positive.

Corollary 4 (A common-objective null). *Suppose the update field of each policy i has the form $\Delta_i = \eta_i P_i \nabla_i Q$, where P_i is symmetric positive definite and depends only on policy-local state. Then*

$$\Omega_{ij} = 0 \qquad \text{for every pair } i, j. \tag{22}$$

Selection can still matter through B_i and I_{ij}, and the finite-batch coverage boundary can still matter because routing coverage need not be preserved by a locally beneficial update.

Corollary 5 (Pairwise reconstruction). *For two programs σ and π on the same selected set $\mathcal{A}$ and the same active update branch,*

$$\Gamma_t^{\mathrm{upd}}(\mathcal{A}, \sigma) - \Gamma_t^{\mathrm{upd}}(\mathcal{A}, \pi) = \tfrac{1}{2} \sum_{\substack{i<j \\ i,j \in \mathcal{A}}} \left(o_{ij}^{\sigma} - o_{ij}^{\pi} \right) \Omega_{ij} + O(\ell^3 \delta^{3/2}). \tag{23}$$

Thus three pair order values reconstruct the smooth relative values of all six full orders in a three-policy system; coverage-feasibility labels are added separately.

Order value on a fixed branch has a physical origin: it is carried by fresh data. Define a stale update of policy j after another policy changes by keeping the stage-start stochastic object, $\Delta_j^{\mathrm{stale}}(\Theta + v) = \mathbb{E}_{\xi \sim P_{\mathsf{G},\Theta}}[h_j(\xi, \Theta + v)]$. For a fresh update, the law of ξ changes as well. When $P_{\mathsf{G},\Theta}$ has a differentiable density, the distribution part of the field derivative is

$$D\Delta_j^{\mathrm{proc}}[v] = \mathrm{Cov}_{P_{\mathsf{G},\Theta}} \left(h_j(\xi, \Theta), \nabla_\Theta \log P_{\mathsf{G},\Theta}(\xi)^\top v \right). \tag{24}$$

Assumption 5 (Conditional policy separation). Given cached ξ and fixed agent-local randomness, the proposal, optimizer state, normalization, clipping, trust-region projection, and certificate of policy i depend only on the trainable state of policy i.

Proposition 3 (Fresh-data pair channel). *Under Assumptions 3 to 5, every stale permutation of the same selected set reaches the same final state. For each pair i, j,*

$$\Omega_{ij} = \left\langle g, D\Delta_j^{\mathrm{proc}}[\Delta_i] - D\Delta_i^{\mathrm{proc}}[\Delta_j] \right\rangle,$$

and

$$I_{ij} = \Delta_i^\top H_Q \Delta_j + \tfrac{1}{2} \left\langle g, D\Delta_j^{\mathrm{proc}}[\Delta_i] + D\Delta_i^{\mathrm{proc}}[\Delta_j] \right\rangle. \tag{25}$$

A nonzero stale order gap falsifies Assumption 5 and reveals hidden shared update state.

The covariance identity in Eq. (24) and its consequences are proved in Appendix B.11.

Corollary 6 (Detectability). *On one active branch, assume uniformly $\Delta_i^\delta = \sqrt{\delta}\, v_i + O(\delta)$ and $D\Delta_i^\delta = \sqrt{\delta}\, Dv_i + O(\delta)$ for fixed fields v_i. Suppose the resulting leading pair coefficient is nonzero. Then*

$$\Omega_{ij} = c_{ij}\delta + O(\delta^{3/2}), \qquad c_{ij} \neq 0, \tag{26}$$

so $|\Omega_{ij}| / \delta$ approaches $|c_{ij}|$. Under Corollary 4, the true ratio is $O(\sqrt{\delta})$ and collapses to zero. If one paired capability difference is sub-Gaussian with variance proxy $2\sigma_{\mathrm{ev}}^2(1-\rho)$, then a paired test resolves the sign with error at most α once

$$n \gtrsim \frac{4\sigma_{\mathrm{ev}}^2 (1-\rho) \log(2/\alpha)}{c_{ij}^2 \delta^2}. \tag{27}$$

B.8 Supporting Lemmas for Ordered Programs

Lemma 3 (Path expansion). *Let $z_0 = z$ and $z_s = z_{s-1} + \Delta_{i_s}(z_{s-1})$ for an ordered program of length ℓ. Then*

$$z_\ell - z = \sum_{s=1}^{\ell} \Delta_{i_s}(z) + \sum_{1 \le r < s \le \ell} D\Delta_{i_s}(z)[\Delta_{i_r}(z)] + R_\ell,$$

where $\|R_\ell\| \le C_{\mathrm{path}} \ell^3 \delta^{3/2}$ for a constant that depends only on c_0, c_1, c_2, and $\|z_s - z\| \le s c_0 \sqrt{\delta}$.

Proof. The displacement bound follows because every field has norm at most $c_0\sqrt{\delta}$ on the declared neighborhood. Let $d_s = z_s - z$. Taylor expansion with integral remainder gives

$$\Delta_{i_s}(z + d_{s-1}) = \Delta_{i_s}(z) + D\Delta_{i_s}(z)[d_{s-1}] + r_s,$$

where

$$\|r_s\| \le \tfrac{1}{2} c_2 \sqrt{\delta}\, \|d_{s-1}\|^2 \le \tfrac{1}{2} c_2 c_0^2 (s-1)^2 \delta^{3/2}.$$

Assume the claimed expansion holds at $s-1$. The first-order pair sum in d_{s-1} has norm at most a constant times $s^2\delta$, so applying $D\Delta_{i_s}$ to it contributes at most a constant times $s^2\delta^{3/2}$. Applying $D\Delta_{i_s}$ to the earlier remainder is absorbed by a cubic induction bound for sufficiently small δ. The only new terms of order at most δ are $D\Delta_{i_s}(z)[\Delta_{i_r}(z)]$ for $r < s$. Adding the new remainder bounds gives $\|R_s\| \le C_{\mathrm{path}} s^3 \delta^{3/2}$ for a fixed large enough constant. The base case is exact, so induction proves the result. □

Lemma 4 (Leading update-noise correction). *Under the quantitative assumptions, the conditional expected capability of a stochastic ordered program equals the capability obtained from the population path plus*

$$\tfrac{1}{2}\delta \sum_{i \in \mathcal{A}} \mathrm{tr}(H_Q \Sigma_i) + O(\ell^3 \delta^{3/2}).$$

The leading covariance correction is independent of the order and is additive over the selected policies.

Proof. Let z_s and $\widehat{z}_s$ be the population and stochastic paths from z, and set $d_s = z_s - z$, $\widehat{d}_s = \widehat{z}_s - z$. The increment bounds and Minkowski's inequality give

$$\left(\mathbb{E}\left\|\widehat{d}_s\right\|^3\right)^{1/3} \le s(c_0 + c_5^{1/3})\sqrt{\delta}.$$

Write $S = \sum_s \Delta_{i_s}(z)$, $N = \sum_s \varepsilon_{i_s}$, and $U = \widehat{d}_\ell - S - N$. The field Lipschitz bound gives $(\mathbb{E}\,\|U\|^2)^{1/2} = O(\ell^2\delta)$, with the same bound for each prefix. Expanding each field at z and taking expectations therefore gives

$$\mathbb{E}\widehat{d}_\ell = S + \sum_{r<s} D\Delta_{i_s}(z)[\Delta_{i_r}(z)] + O(\ell^3\delta^{3/2}) = d_\ell + O(\ell^3\delta^{3/2}).$$

The linear noise terms vanish. Taylor remainders contribute $O(\sqrt{\delta}\sum_s \mathbb{E}\left\|\widehat{d}_{s-1}\right\|^2)$, and substituting the prefix expansions contributes $O(\sqrt{\delta}\sum_s s^2\delta)$. The last equality uses Lemma 3.

Set $H = \nabla^2 Q(z)$. Zero noise cross-covariances give $\mathbb{E}[N^\top H N] = \delta \sum_s \mathbb{E}\,\mathrm{tr}(H\Sigma_{i_s}(\widehat{z}_{s-1}))$. Lipschitz continuity of Σ_i and the displacement bound replace the random states here by z with error $O(\ell^2\delta^{3/2})$. Using $\mathbb{E}N = 0$ and the bound on U, Cauchy–Schwarz gives

$$\mathbb{E}[\widehat{d}_\ell^\top H \widehat{d}_\ell] = d_\ell^\top H d_\ell + \delta \sum_{i \in \mathcal{A}} \mathrm{tr}(H\Sigma_i(z)) + O(\ell^3\delta^{3/2}).$$

The terms involving U have this order because $\ell\sqrt{\delta}$ is bounded; replacing $S^\top H S$ by $d_\ell^\top H d_\ell$ has the same cost. Taylor expansion of Q at z is valid along both paths in the declared neighborhood. The third-moment bound makes its expected remainder $O(\ell^3\delta^{3/2})$. Substituting the two moment identities proves the stated covariance correction. □

B.9 Proof of Theorem 5

Proof. First, ignore update noise and write all fields at the stage-start state unless an argument is shown. By Lemma 3, the deterministic path displacement is

$$d_\sigma = S_\sigma + T_\sigma + R_\sigma,$$

where

$$S_\sigma = \sum_{r=1}^{\ell} \Delta_{i_r}, \qquad T_\sigma = \sum_{1 \le r < s \le \ell} D\Delta_{i_s}[\Delta_{i_r}],$$

$\|S_\sigma\| = O(\ell\sqrt{\delta})$, $\|T_\sigma\| = O(\ell^2\delta)$, and $\|R_\sigma\| = O(\ell^3\delta^{3/2})$. A third-order expansion of Q around Θ gives

$$Q(\Theta + d_\sigma) - Q(\Theta) = \langle g, d_\sigma \rangle + \tfrac{1}{2} d_\sigma^\top H_Q d_\sigma + \varrho_\sigma,$$

with $|\varrho_\sigma| \le M_3 \|d_\sigma\|^3 / 6 = O(\ell^3\delta^{3/2})$. Because $S_\sigma^\top H_Q T_\sigma = O(\ell^3\delta^{3/2})$ and all terms containing R_σ are of the same or smaller order,

$$Q(\Theta + d_\sigma) - Q(\Theta) = \langle g, S_\sigma \rangle + \langle g, T_\sigma \rangle + \tfrac{1}{2} S_\sigma^\top H_Q S_\sigma + O(\ell^3\delta^{3/2}).$$

The one-policy terms are

$$\sum_{i \in \mathcal{A}} \left(\langle g, \Delta_i \rangle + \tfrac{1}{2} \Delta_i^\top H_Q \Delta_i \right).$$

For one pair i, j, if i precedes j, its contribution is

$$\Delta_i^\top H_Q \Delta_j + \langle g, D\Delta_j[\Delta_i] \rangle = I_{ij} + \tfrac{1}{2}\Omega_{ij}.$$

If j precedes i, the contribution is $I_{ij} - \frac{1}{2}\Omega_{ij}$. Lemma 4 adds the covariance term in Eq. (17) and changes only the uniform remainder. Combining path, value, and noise constants gives Eq. (20). □

B.10 Proofs of Update Corollaries

Proof of Corollary 3. Apply Theorem 5 to u and $u \oplus_p i$. Every term involving only policies in $\mathcal{A}$ cancels. The remaining leading terms are B_i and one ordered pair term between i and each $j \in \mathcal{A}$, which gives Eq. (21). The two theorem remainders give a uniform insertion remainder of order $(|\mathcal{A}| + 1)^3\delta^{3/2}$. The sign conclusions compare the leading margin with this remainder. The coverage-boundary qualification follows from conditioning on a fixed active branch. □

Proof of Corollary 4. Write the gradient in per-policy blocks as $g = (g_1, \ldots, g_L)$ and let $H_{ji} = \nabla^2_{ji} Q$, so $H_{ij} = H_{ji}^\top$. For $i \neq j$,

$$\Delta_i = \eta_i P_i g_i, \qquad \Delta_j = \eta_j P_j g_j.$$

Because P_j depends only on policy j, it does not change in direction Δ_i. Therefore

$$\langle g, D\Delta_j[\Delta_i] \rangle = \eta_i \eta_j g_j^\top P_j H_{ji} P_i g_i.$$

Similarly,

$$\langle g, D\Delta_i[\Delta_j] \rangle = \eta_i \eta_j g_i^\top P_i H_{ij} P_j g_j.$$

The two scalars are transposes of each other because P_i, P_j are symmetric and $H_{ij} = H_{ji}^\top$. They are equal, so Eq. (19) is zero. The argument does not remove the direct, symmetric, or finite-batch coverage terms. □

Proof of Corollary 5. Apply Eq. (20) to σ and π. The selected set is the same, so every direct and symmetric selection term cancels. The remaining order terms give Eq. (23); the difference of the uniform remainders remains $O(\ell^3\delta^{3/2})$. □

Proof of Corollary 6. Under the quantitative assumptions, write

$$\Delta_i^\delta = \sqrt{\delta}\, v_i + O(\delta), \qquad D\Delta_i^\delta = \sqrt{\delta}\, D v_i + O(\delta)$$

uniformly on the declared neighborhood. Substitution into Eq. (19) gives

$$\Omega_{ij} = \delta \langle g, Dv_j[v_i] - Dv_i[v_j] \rangle + O(\delta^{3/2}).$$

The coefficient is c_{ij}. If it is nonzero, division by δ proves the plateau. Under Corollary 4, the coefficient is zero, so the ratio is $O(\sqrt{\delta})$.

Let $Z_1, \dots, Z_n$ be paired capability differences with mean Ω_{ij} and sub-Gaussian variance proxy $2\sigma_{\text{ev}}^2(1-\rho)$. Their mean satisfies

$$\Pr\left(\left|\overline{Z} - \Omega_{ij}\right| \geq |\Omega_{ij}|\right) \leq 2\exp\left(-\frac{n\Omega_{ij}^2}{4\sigma_{\text{ev}}^2(1-\rho)}\right).$$

Making this at most α and using $\Omega_{ij} = c_{ij}\delta(1 + O(\sqrt{\delta}))$ gives Eq. (27), up to the conventional constant absorbed by $\gtrsim$. □

B.11 Proof of Proposition 3

Proof. Fix a cached stochastic object ξ and all agent-local randomness. Under Assumption 5, the complete update map for policy i can be written as

$$\Phi_i^\xi(z) = (z_1, \dots, \phi_i^\xi(z_i), \dots, z_L),$$

where z_i contains the policy parameters and all policy-local update state. For $i \neq j$, the maps act on disjoint coordinates, so $\Phi_i^\xi \circ \Phi_j^\xi = \Phi_j^\xi \circ \Phi_i^\xi$ pointwise. Every stale permutation of a selected set therefore reaches the same final state.

For the fresh population field, write

$$\Delta_j(\Theta) = \int h_j(\xi, \Theta_j) p_\Theta(\xi)\, d\xi,$$

where separation makes the integrand depend only on the local policy state. Differentiating in direction v gives

$$\begin{aligned} D\Delta_j[v] = &\int D_{\Theta_j} h_j(\xi, \Theta_j)[v_j] p_\Theta(\xi)\, d\xi \\ &+ \int h_j(\xi, \Theta_j) p_\Theta(\xi) \nabla_\Theta \log p_\Theta(\xi)^\top v\, d\xi. \end{aligned}$$

For $v = \Delta_i$ with $i \neq j$, the local component v_j is zero, so the first integral vanishes. The score has mean zero, which turns the second integral into the covariance in Eq. (24). Substitution into Eqs. (18) and (19) proves the two identities. A nonzero stale order gap contradicts exact commutation, so at least one part of Assumption 5 fails. □

C LACA DETAILS

C.1 IMPLEMENTATION DETAILS

Algorithm 1: LACA: one decision window.

Input: $\mathcal{F}_t$, ownership map b_t, prescribed seed, budgets, audit tasks.
Sort created nodes by stable ID; sample at most three without replacement using the seed;
Form $\mathcal{P}_t$ from STOP, all singleton updates, and all ordered pairs of distinct sampled nodes;
On temporary copies, execute non-STOP candidates and their common downstream continuation, sharing trained prefix checkpoints and collecting to m_i effective groups within each owner's raw-group allocation;
Form $\mathcal{P}_t^{\mathrm{ok}}$ from completed, budget-feasible trials and STOP;
Estimate mean two-stage gains against STOP on paired development tasks;
$u^\star \leftarrow \arg\max_{u \in \mathcal{P}_t^{\mathrm{ok}}} \widehat{\Gamma}_t^{(2)}(u)$, favoring shorter programs and STOP at zero-gain ties;
if $u^\star = \text{STOP}$ **then**
    Charge all incurred costs to $\mathcal{F}_t$; **return** $\mathcal{F}_t$;
$(\mathcal{A}^\star, \sigma^\star) \leftarrow u^\star$; copy $\mathcal{F}_t' \leftarrow \mathcal{F}_t$;
for *owner* i *in* $\sigma^\star$ **do**
    Reset $N_i^{\mathrm{eff}} \leftarrow 0$; collect fresh natural or controlled groups under $\mathcal{F}_t'$ to final root rewards until $N_i^{\mathrm{eff}} \geq m_i$ or the allocated group budget is exhausted;
    if $N_i^{\mathrm{eff}} < m_i$ **then**
        Charge incurred costs to $\mathcal{F}_t$; **return** $\mathcal{F}_t$;
    On i's first update, copy its SFT adapter into a private LoRA and create its optimizer;
    Update i on its own response tokens; retain its state and discard the groups;
Audit the completed current program $\mathcal{F}_t'$ against $\mathcal{F}_t$ on paired root tasks;
$\mathcal{F}_{\mathrm{next}} \leftarrow \mathcal{F}_t$;
if *team gain lower bound is positive and resource limits hold* **then**
    $\mathcal{F}_{\mathrm{next}} \leftarrow \mathcal{F}_t'$;
Monitor $\mathcal{F}_t'$ against $\mathcal{F}_t$ on separate paired tasks, keeping $\mathcal{F}_{\mathrm{next}}$ fixed;
Charge all incurred costs, including candidate training, selection, re-execution, audit, and monitoring, to $\mathcal{F}_{\mathrm{next}}$; **return** $\mathcal{F}_{\mathrm{next}}$;

C.1.1 POLICY OWNERSHIP, LOGGING, AND UPDATES

The benchmark implementation assigns each runtime node its own update owner. The root uses a frozen Qwen3.5-9B backbone; worker nodes share a frozen Qwen3.5-4B backbone. Before its first update, a node uses the common SFT policy for its role. Its first update copies that adapter into a private LoRA and initializes a separate optimizer state. Subsequent updates continue from the node's own parameters and optimizer state. Update counters, normalization statistics, and random seed streams are also owner-local. The shared-state controlled variant maps both worker call sites to one owner. Independent optimizer clocks preserve the policy-local state in Assumption 5.

Each root rollout group records activation and completion by owner, the reward vector and within-group variance, the token mask after clipping and length filtering, and the count of tokens contributing a nonzero policy-gradient term. Candidate logs also retain $Y_{n,c}$ whenever the candidate call completes. These records distinguish actor eligibility from the gradient produced by an update.

Role-matched SFT or OPD supplies owner-local bootstrap targets. Selector examples cover solve, delegate, and create choices with paired final team outcomes; worker examples target their own responses. Sequential recollection and the complete-program audit apply to both training and temporary branches.

C.1.2 CANDIDATE SPECIFICATION AND BOOTSTRAP INPUTS

A candidate trainable policy is a fixed record containing its parent, role description, tools, context interface, initialization, adapter shape, controlled exposure, effective-group threshold, update budget, and separate evaluation protocol. Bootstrap curricula may contain fixed create demonstrations when the selector cannot yet reach the action. Still, the deployed selector proposes the role, parent, context, and tools from the root task. The training system fills the remaining fields before candidate training,

and the record cannot change after the audit begins. The controlled studies allow at most one candidate per parent and commit at most one persistent policy creation per audit. Creating a runtime call does not itself create or promote a new policy. The recursive harness can execute multiple child calls; its policy ownership and training configuration are specified in Appendices D.1 and D.4.

C.2 Coverage Estimation and Controller Guarantees

The following analysis gives sufficient conditions for coverage feasibility and commitment.

C.2.1 Coverage and Evidence Estimates

The variables used by the controller are directly available in the training log: for policy i and an audit window of N root units, define

$$\widehat{s}_i = \frac{1}{N}\sum_{n=1}^{N} Z_{n,i}, \qquad N_i^{\mathrm{eff}} = \sum_{n=1}^{N} Z_{n,i}. \tag{28}$$

Proposition 4 (Update-coverage identity)**.** *For every policy i,*

$$\mathbb{E}[N_i^{\mathrm{eff}}] = N s_i, \qquad s_i = \rho_i f_i v_i \nu_i. \tag{29}$$

Each factor is identified by its conditional frequency in the rollout log when its conditioning event has positive probability. For a candidate at a fixed depth-d site, reach is the product of the conditional routing rates on its path. An owner shared by multiple sites uses the probability of the union of their activations.

The factors diagnose where groups are lost; the controller constructs coverage bounds directly from the joint indicators $Z_{n,i}$.

Lemma 5 (Simultaneous coverage interval)**.** *For L policies and N conditionally independent root groups, define*

$$L_i^s = \left[\widehat{s}_i - \sqrt{\frac{\log(2L/\alpha)}{2N}}\right]_+, \tag{30}$$

and

$$U_i^s = \min\left\{1, \widehat{s}_i + \sqrt{\frac{\log(2L/\alpha)}{2N}}\right\}. \tag{31}$$

Then all true coverage values lie in $[L_i^s, U_i^s]$ simultaneously with probability at least $1-\alpha$.

For candidate c, let L_c^{ev} be the lower confidence bound on its audit exposure s_c^{ev}, obtained from the logged activation and completion events by the same construction. A conservative estimate of the passive root-group requirement is

$$\widehat{N}_c^{\mathrm{id,pass}} = \max\left\{\frac{8\log(4/\alpha)}{L_c^{\mathrm{ev}}}, \frac{4\widehat{\sigma}_c^2 \log(4/\alpha)}{L_c^{\mathrm{ev}}\epsilon^2}\right\}, \tag{32}$$

with the convention that it is infinite when $L_c^{\mathrm{ev}} = 0$; here $\widehat{\sigma}_c^2$ is an upper confidence bound on the evidence variance proxy. The controlled requirement replaces L_c^{ev} by a lower bound on $e_c f_c$.

Here B_{rem} and B_{ctl} count the root groups affordable under the remaining and allocated rollout budgets, respectively, using each protocol's charged rollouts per group for conversion.

Lemma 6 (Remaining-budget evidence rule)**.** *If $\widehat{N}_c^{\mathrm{id,pass}} > B_{\mathrm{rem}}$, the declared sufficient condition cannot be certified from the current exposure interval within the remaining root-group allocation. If a controlled protocol has requirement $\widehat{N}_c^{\mathrm{id,ctl}} \le B_{\mathrm{ctl}}$, then the declared precision and confidence are feasible within its allocated controlled budget. If neither requirement fits, the correct output is insufficient evidence, not a negative candidate value.*

C.2.2 Coverage and Budget Feasibility

The following local conditions characterize a coverage-certified controller. The benchmark collection rule in Appendix D.4 uses observed effective-group counts; Eq. (33) supplies the sufficient coverage condition for the analysis below.

Assumption 6 (Valid simultaneous intervals)**.** With probability at least $1-\alpha$, every capability gain, coverage, audit exposure, coverage multiplier, variance proxy, and declared audit outcome used by one decision lies in its reported simultaneous interval, including sampling, proxy, and finite-branch error.

Assumption 7 (Local coverage transfer)**.** Within one decision stage and the stage that follows it, the predicted joint coverage multiplier of each evaluated commit set differs from its realized value by at most the reported interval radius. After a graph change, all coverage and capability models are recalibrated before another audit decision.

Assumption 8 (Fixed budget accounting)**.** Every compared update program uses the same per-update root rollout budget and the same separate evaluation tasks. Every candidate comparison uses a declared passive or controlled root rollout budget and fixed inference limits. The accounting does not convert tokens, optimizer steps, accelerator time, wall time, or evaluation inference latency into a scalar opportunity cost.

Let m_c be the minimum effective groups required by downstream policy c in a stage of B root groups and let $L_t^s(c \mid u)$ be a lower confidence bound on its coverage after ordered program u has committed the updates that precede c. Because later commits can also change the coverage that c receives in the next stage, let $L_{t+1}^s(c \mid u)$ be the corresponding lower bound after all updates in u have been committed, matching the reference continuation of Eq. (1). Let $\mathcal{D}(u)$ be the downstream policies selected by u that require a coverage certificate and $k(u) = 2\,|\mathcal{D}(u)|$ the number of policy-stage tests the program runs. A sufficient program-level coverage-feasibility test is

$$\min\{L_t^s(c \mid u), L_{t+1}^s(c \mid u)\} \geq \frac{m_c}{B} + \sqrt{\frac{\log(k(u)/\alpha)}{2B}} \qquad \text{for every } c \in \mathcal{D}(u). \tag{33}$$

This is a conditional certificate for a fixed entry pool and declared sampling schedule. At each tested stage, fix the prior history, including interval data, current parameters, and the selected program, then draw B fresh, conditionally independent groups. Coverage bounds refer to this fixed-pool sampling law; variation across newly sampled parent states is a separate source of uncertainty. For each such history with valid bounds under Assumption 6, Hoeffding's inequality bounds the probability of fewer than m_c effective groups by $\alpha/k(u)$. A union bound over the $k(u)$ tests gives failure probability at most α under these stagewise conditions for a program that passes Eq. (33). The test is demanding at small budgets: with $m_c = 4$, one certified downstream policy ($k(u) = 2$), and $\alpha = 0.05$, the right-hand side is 1.22 at $B = 6$, 0.98 at $B = 8$, and 0.47 at $B = 22$, so no program can be certified at $B = 6$. A passive candidate reaches its evidence deadline when

$$\widehat{N}_c^{\mathrm{id,pass}} > B_{\mathrm{rem}}, \qquad \widehat{N}_c^{\mathrm{id,ctl}} \leq B_{\mathrm{ctl}}. \tag{34}$$

Under the current intervals, the first inequality leaves passive identification uncertified within the remaining budget; the second certifies controlled identification at the declared precision. Here B_{ctl} is reserved from the same stage and audit root budget as B_{rem}, so controlled activation reallocates root groups instead of adding data.

C.2.3 Team Utility and Commitment

Commitment of an update program. The audit pairs the completed current program $\mathcal{F}_t'$ with the active team $\mathcal{F}_t$ on root tasks, matching seeds and inference limits. For audit task q, the paired reward difference is $d_{t,q}^{\mathrm{aud}} = r_q(\mathcal{F}_t') - r_q(\mathcal{F}_t)$, where r_q is the final root-task reward. Let $L_{t,Q}^{\mathrm{aud}}$ be a lower confidence bound on the mean of these paired differences. For each capability guardrail k in the declared set $\mathcal{K}_t$, let $L_{t,k}^{\mathrm{aud}}$ be its paired-gain lower bound and $\epsilon_k \geq 0$ its allowed regression. The commitment rule is

$$L_{t,Q}^{\mathrm{aud}} > 0 \quad \text{and} \quad L_{t,k}^{\mathrm{aud}} \geq -\epsilon_k \quad \text{for every } k \in \mathcal{K}_t.$$

A passing audit commits all selected owner updates together; otherwise, the active team is retained. The audit evaluates the current program; the common downstream continuation enters the two-stage

comparison used for program selection. Audit trajectories count toward the same rollout budget as probes, selection, and training. The benchmark instantiation uses the team-utility bound and hard execution limits, with $\mathcal{K}_t = \varnothing$; its audit sample size and bootstrap settings are in Appendix D.4. Additional capability coordinates enter the persistent-policy expansion rule below.

Program and graph sufficiency. At an update stage, the current parameters are sufficient on the evaluated program set when

$$\max_{u \in \mathcal{P}_t \setminus \{\textsc{Stop}\}} U_t^Q(u) \le 0, \tag{35}$$

where $U_t^Q(u)$ is an upper confidence bound on the two-stage value $\Gamma_t^{(2)}(u; \omega_t)$ of Eq. (1). At a candidate audit, the current graph is sufficient on the evaluated candidate set when

$$\max_{c \in \mathcal{C}_t} U_{t,c}^{\mathrm{lh}} \le 0, \tag{36}$$

in which case the audit returns STAY, the decision to retain the current delegation graph without committing a candidate. For each candidate, the interval is measured after its fixed training protocol as a paired difference between the complete candidate system and the current active system on the same root audit tasks under the declared rollout budget and inference limits. The three coordinates therefore measure final team capability, not the candidate's local response quality. A candidate $c^\star$ is committed when

$$L_{t,c^\star}^{\mathrm{lh}} > 0, \qquad L_{t,c^\star}^{\mathrm{gen}} \ge -\epsilon_{\mathrm{gen}}, \qquad L_{t,c^\star}^{\mathrm{mt}} \ge -\epsilon_{\mathrm{mt}}, \tag{37}$$

and its long-horizon lower bound exceeds the upper bounds of the other evaluated candidates. The current delegation graph is locally sufficient when both sufficiency tests hold:

$$\max\left\{ \max_{u \neq \textsc{Stop}} U_t^Q(u), \max_{c \in \mathcal{C}_t} U_{t,c}^{\mathrm{lh}} \right\} \le 0. \tag{38}$$

C.2.4 LOCAL CONTROLLER GUARANTEE

Theorem 6 (Coverage-aware intervention rule)**.** *Under Assumptions 1 to 8, on the simultaneous confidence event and on the declared update and candidate sets, the following claims hold. First, if Eq. (35) holds,* STOP *is optimal among the evaluated update programs. Otherwise, maximizing* $\widehat{\Gamma}_t^{(2)}$ *over programs satisfying Eq. (33) selects a program whose true two-stage value is within twice the largest gain-interval radius for* Q^{aud} *of the best value in that certified set under the reference continuation. Second, a program that violates Eq. (33) cannot be certified to supply the required downstream effective groups; LACA may restore feasibility by moving the downstream update earlier, removing some upstream updates, or assigning declared controlled groups. Theorem 2 shows that this exclusion can reject update-all even when every singleton update gain is positive. Third, Eq. (34) certifies that passive feasibility cannot be established from the current intervals while a controlled parent-state audit remains certified feasible. Fourth, Eq. (36) makes* STAY *optimal among the evaluated candidates, while Eq. (37) selects a candidate with a positive paired long-horizon team-gain lower bound and the declared general and multi-turn team guardrails. Consequently, Eq. (38) certifies that no evaluated update program has a positive upper bound on its two-stage value and no evaluated candidate system has a positive target team-gain upper bound under the declared budgets.*

The proof uses interval ordering for update and candidate decisions, Hoeffding concentration for the coverage-feasibility test, and Theorem 1 for the passive deadline. The proof is in Appendix C.4.

Remark 4 (Scope of the guarantee). The result is local to the visited and calibrated states, the declared candidate protocols, the evaluated finite action sets, and the fixed root rollout and inference budgets. The guarantee compares ordered subsets screened by Eq. (33) under the fixed continuation in Eq. (1). A complete program refreshes training data after each owner update. If another decision window is run after its complete-team audit, that selection has its own local comparison. The guarantee concerns the evaluated feasible set, not a universal update order or global optimality across decision windows.

C.3 Proofs for Coverage Estimation

Proof of Proposition 4. By definition, $N_i^{\text{eff}} = \sum_{n=1}^{N} Z_{n,i}$ and $\mathbb{E}[Z_{n,i}] = s_i$. Linearity of expectation gives

$$\mathbb{E}[N_i^{\text{eff}}] = \sum_{n=1}^{N} \mathbb{E}[Z_{n,i}] = N s_i.$$

The chain rule for the nested events activation, finish, nonzero advantage, and usable tokens gives

$$s_i = \Pr(A_{n,i} = 1) \Pr(F_{n,i} = 1 \mid A_{n,i} = 1)$$
$$\times \Pr(V_{n,i} = 1 \mid A_{n,i} F_{n,i} = 1) \Pr(W_{n,i} = 1 \mid A_{n,i} F_{n,i} V_{n,i} = 1),$$

which is $\rho_i f_i v_i \nu_i$. Each factor is the limit of its empirical conditional frequency. For a depth-d candidate, activation requires every path edge, so its activation probability is the product of the recorded conditional routing rates. □

Proof of Lemma 5. For one policy, Hoeffding's inequality gives

$$\Pr\left(|\widehat{s}_i - s_i| > \sqrt{\frac{\log(2L/\alpha)}{2N}} \right) \le \frac{\alpha}{L}.$$

A union bound over the L policies shows that all unclipped intervals hold with probability at least $1 - \alpha$. Clipping a lower endpoint at zero and an upper endpoint at one preserves coverage because every coverage value is a probability. □

Proof of Lemma 6. Equation (32) replaces audit exposure with its lower confidence bound. If the resulting requirement exceeds B_{rem}, the sufficient condition of Theorem 1 cannot be certified within the passive budget. A controlled requirement no larger than B_{ctl} fits its allocated budget. If neither fits, the audit returns insufficient evidence at the declared precision and confidence. □

C.4 Proof of Theorem 6

Proof. Let $\mathcal{E}$ be the simultaneous confidence event in Assumption 6. The interval comparisons hold on $\mathcal{E}$. Coverage probabilities use fresh groups conditional on the pre-stage history, as specified below.

For update sufficiency, Eq. (35) implies

$$\Gamma_t^{(2)}(u; \omega_t) \le U_t^Q(u) \le 0$$

for every evaluated nonempty program. The empty program STOP has two-stage value zero by Eq. (1), so it is optimal. Otherwise let $u^\star$ maximize $\Gamma_t^{(2)}(\cdot; \omega_t)$ among programs satisfying Eq. (33), and let $\widehat{u}$ maximize the estimate $\widehat{\Gamma}_t^{(2)}$ on the same feasible set; write $\Gamma(u)$ for $\Gamma_t^{(2)}(u; \omega_t)$ and $\widehat{\Gamma}(u)$ for its estimate $\widehat{\Gamma}_t^{(2)}(u)$. On $\mathcal{E}$ every evaluated $\Gamma(u)$ lies in its reported Q^{aud} interval; let ε_t be the largest radius of these intervals over the certified set, so that $\left|\widehat{\Gamma}(u) - \Gamma(u)\right| \le \varepsilon_t$ for every such u. Then

$$\begin{aligned}\Gamma(u^\star) - \Gamma(\widehat{u}) &\le \Gamma(u^\star) - \widehat{\Gamma}(u^\star) \\ &\quad + \widehat{\Gamma}(\widehat{u}) - \Gamma(\widehat{u}) \\ &\le 2\varepsilon_t,\end{aligned}$$

where the omitted estimated difference is nonpositive because $\widehat{u}$ maximizes the estimates.

Now fix a policy $c \in \mathcal{D}(u)$ and the tested stage's prior history, including its entry pool, sampling schedule, current parameters, interval data, and selected program. The B subsequent groups are conditionally independent under this sampling law; all probabilities below are conditional on that history. For histories with valid bounds on $\mathcal{E}$, its true post-prefix coverage is at least $L_t^s(c \mid u)$, and by Assumption 7 its true next-stage coverage under the reference continuation ω_t, which adds no

upstream commit, is at least $L^s_{t+1}(c \mid u)$; the argument below applies to either bound. Let K_c be its effective-group count among the B root groups of the tested stage. If Eq. (33) holds, then

$$s_c - \frac{m_c}{B} \geq \sqrt{\frac{\log(k(u)/\alpha)}{2B}}.$$

Hoeffding's inequality gives

$$\Pr(K_c < m_c) = \Pr\left(\frac{K_c}{B} - s_c < \frac{m_c}{B} - s_c\right) \leq \exp\left(-2B\left(s_c - \frac{m_c}{B}\right)^2\right) \leq \frac{\alpha}{k(u)},$$

and a union bound over the $k(u)$ policy-stage tests bounds the probability that any declared downstream update of u misses its target by α. A program that fails the test lacks this certificate. Moving c earlier shrinks the set of commits in effect in the current stage only. Removing upstream updates shrinks that set in both stages. Each change raises the joint multiplier. Controlled groups add a declared positive exposure. These are the theorem's three repairs. Theorem 2 gives a regime where update-all fails this coverage test despite positive singleton marginals.

The passive evidence statement is Lemma 6 together with Theorem 1. Equation (34) says the passive sufficient requirement no longer fits in B_{rem} while the controlled sufficient requirement fits in B_{ctl}. Thus the rule changes the evidence source before the passive budget expires.

For audit sufficiency, the current delegation graph has candidate gain zero by definition of the paired difference. If Eq. (36) holds, every evaluated candidate has true long-horizon gain no larger than its nonpositive upper bound, so STAY is optimal on the evaluated set. If Eq. (37) holds, the complete system containing candidate $c^\star$ has positive true paired long-horizon team gain and satisfies the two team-level no-regression guardrails on $\mathcal{E}$. If its lower bound also exceeds every other candidate upper bound, it is optimal among the evaluated candidates and STAY.

Finally, Eq. (38) is the conjunction of update and audit sufficiency. The first part makes STOP optimal for the current stage, and the second makes STAY optimal among the candidate subagents under their fixed budgets. This proves the final claim. □

C.5 PROGRAM ESTIMATES AND CALIBRATION

The controller needs estimates of program value and coverage at the current state. Each evaluated program declares its owner subset, order, per-update budget, continuation, and natural or controlled entries. Paired temporary branches estimate its team gain against STOP on development tasks. The comparison includes the declared continuation in Eq. (1), so it measures team value across both learning stages.

For the controller guarantee, one construction scales capability to $[0, 1]$, fixes M programs at the current state, and collects n independent complete training and evaluation replicates per program. Each replicate pairs the program with STOP, giving a gain $D_{u,k} \in [-1, 1]$ with mean $\Gamma^{(2)}_t(u; \omega_t)$. For $\widehat{\Gamma}(u) = n^{-1} \sum_k D_{u,k}$, Hoeffding's inequality and a union bound give simultaneous intervals $[\widehat{\Gamma}(u) - r, \widehat{\Gamma}(u) + r]$, where $r = \sqrt{2\log(2M/\alpha_Q)/n}$, with probability at least $1 - \alpha_Q$. Assigning STOP its exact value zero, maximizing these estimates incurs at most $2r$ loss relative to the best program in the same certified set. The error budget α_Q is allocated from the total confidence budget; coverage transfer and repeated comparisons follow Assumptions 6 and 7.

The local decomposition in Appendix B.7 separates direct values B_i, symmetric interactions I_{ij}, and order terms Ω_{ij}. These quantities explain program rankings; LACA selects programs using the paired value estimates. The coverage estimate records $\rho_i, f_i, \upsilon_i, \nu_i$ and the changes after preceding updates. Joint changes are measured instead of assumed to equal products of single-update multipliers. A positive capability estimate can still fail the coverage condition.

The evaluated program set is finite and declared before its comparisons. Diagnostics and temporary branches use the selection budget, and unused programs are not treated as evaluated alternatives. Program ranking uses development evidence, the final gate compares the selected complete team, and public tasks supply the final test. The guarantee in Theorem 6 requires simultaneous intervals that include sampling and estimation error. Fixed-checkpoint task bootstrap measures evaluation uncertainty; training randomness and adaptive reuse require the corresponding joint error control.

Calibrating predicted against realized coverage, program rankings, gain signs, and interval coverage requires separate complete training replicates.

C.6 Fixed-Budget and Compute Accounting

The update-combination and program-library studies fix $B = 6, 8, 22$ groups per owner update; total collection cost varies with the number of updates. They pair task identities and random seeds and use a shared separate evaluation batch. Every candidate comparison declares its passive and controlled root rollout budgets before outcomes are observed.

Benchmark RL methods share a 2,048-rollout cap. Baseline GRPO/DAPO training uses eight tasks with 32 trajectories per task and a configured maximum of 500 steps. LACA runs one decision window with eight trajectories per group. Its phase allocations total at most 1,792 executions, including temporary candidate training, selection, re-execution, audit, and independent monitoring (Table 6). The remaining 256 executions are unallocated. Natural and controlled trajectories, rejected trials, and DAPO's discarded uniform-reward groups all count toward the cap. Each execution is one sampled team trajectory through its final root outcome. The separate repeated-window component and worker-training studies use checkpoints at 512, 1,024, 1,536, and 2,048 charged rollouts. Shared training prefixes are charged once when reporting the physical cost of a program library; per-program costs include the prefixes used by that program.

D Experimental Details

This appendix specifies policy ownership, training settings, and the evaluation protocols for signal access, program selection, and team capability.

Table 3 links the comparisons to their run identifiers, evaluation units, and budgets. Benchmark methods and component comparisons share the 928-target SFT initialization. For the fixed-program and rollback comparisons, W1 and W2 are the first two nodes in stable-ID order among all created non-root nodes at the start of each run's decision window. At both natural and controlled entry, the fixed reference updates $R \to W_1 \to W_2$ under the 2,048-rollout cap; LACA selects programs of at most two owners from its sampled pool. The program-library study uses 6 separate initial teams with three training repeats. Owner rollback reuses the final LACA checkpoints, pairing benchmark, run, checkpoint, and evaluation record.

The full-LACA row in Table 1(A) and the rollback interventions use each main LACA (GRPO) run's final checkpoint under the 2,048-rollout cap. The objective and gate diagnostics in Appendix E.4 use a separate repeated-window experiment. For these comparisons, run records contain run identifiers, training seeds, checkpoints, charged rollouts by controller phase, and evaluation task sets. For paired score differences d_r, nominal 95% t intervals use $\bar{d} \pm t_{0.975,n-1} s_d/\sqrt{n}$, with $n = 5$ training seeds for the method contrasts in Table 10 and Figure 18(a), and $n = 18$ state–repeat pairs at each budget for Figure 15. The configuration contrasts in Table 17, rollback losses in Table 7, and per-team differences in Figure 16 use paired task $\times$ repeat percentile bootstrap intervals. For the continuation comparison, Table 15 aggregates the six team-level differences and uses a 95% t interval across those six teams. Figure 4 uses half-widths $t_{0.975,4} s/\sqrt{5}$, with s the standard deviation across the five runs. The combination test in Figure 14 uses one-sided simultaneous bounds, max-standardized over all 144 program contrasts before classifying cells by their observed outcome patterns, conditional on the fixed evaluation paths.

D.1 Harness and Policy Ownership

The harness extends SearchSwarm's delegation-and-return design (Lan et al., 2026) to recursive execution. Each node has a local instruction, observations, available tools, and a parent. The root uses DELEGATE and a non-leaf child uses CREATE; both instantiate a child call with their assigned task. A RETURN action supplies a result to the waiting parent. Calls execute serially: the parent resumes its saved context, checks and integrates the child's return, and may execute tools, delegate again, or return. Model parameters remain fixed throughout each inference trajectory. Terminal actions

Table 3: Experimental design and recorded identifiers. Each evaluation task contributes one reward per run and checkpoint.

Evidence	Initial teams and runs	Learner and evaluation	Training budget
Fig. 4	Common SFT: 209 root and 719 worker targets	GRPO, DAPO, Dr.MAS, and two LACA variants; three benchmarks	2,048-rollout cap; LACA: one window, $G = 8$; GRPO/DAPO: 8×32 per batch; Dr.MAS: published settings
Figs. 5, 9	`P7_common_sft`; five repeats, 64 groups per benchmark and repeat	Natural/controlled entry; site and owner coverage under both ownership maps	target depth 0–6; no parameter updates
Fig. 14	`P01`–`P03`, `N01`–`N03`; three repeats per condition and budget	Two single additions, six full orders; 256 evaluation tasks; 144 contrasts	6, 8, 22 groups per owner update
Figs. 6(b), 15	Six starting teams; three training repeats per team and budget	GRPO, $m = 4$; sixteen programs; 32 development and 256 evaluation tasks; second-stage entry intervention	6, 8, 22 groups per owner update
Table 1; Fig. 18	Common SFT; five paired seeds	Entry–selection factorial and rollback; separate repeated-window objective/gate and worker-training studies	2,048-rollout cap for main checkpoints; 512–2,048 rollouts in repeated-window studies
Case 1 in Appendix E.6	Controlled QA checkpoints before and after upstream updates	MuSiQue routing trace and subsequent child training	11 groups per child continuation
Cases 2–3 in Appendix E.6	QA-trained LACA-DAPO and its common SFT initialization	Four Nemotron development tasks; worker calls share one adapter	Fixed-weight inference: depth six, 64 actions, 2,048 tokens per call

use Harbor's environment interface.[1] The runtime records the actual messages sent to the model, generated token IDs, call ancestry, tool observations, and owner of every response.

D.2 Data Construction

D.2.1 Controlled QA Curricula and Saved-State Entries

The controlled QA study uses MuSiQue questions with dependent subquestions, intermediate answers, and supporting paragraphs (Trivedi et al., 2022). The local source contains 19,938 training questions. The QA curriculum selects 705 source questions and contains 705 root, 570 child-1, and 1,651 child-2 entries, for 2,926 entries in total. The protocol bridge contains 16 scripted reference trajectories and 152 role-specific target actions. The recursive harness represents this curriculum as response-only targets and saved-state entries.

Each entry follows the question's dependency order. An unresolved subquestion becomes a delegated contract; its return is inserted into the waiting parent's context, which continues the remaining reasoning. A response-only SFT example contains the owner's current messages and one reference response. An RL entry instead contains the source task, owner, saved parent stack, observations, and remaining budget. After entering the saved state, the current team generates all actions until the root finishes, and the original task evaluator supplies the reward. For terminal entries, replay also recreates files and environment state by executing the prefix in a fresh container.

For example, source question `3hop1__101723_387516_145746` asks which city Nicholas I died in, identifying him through Albert and Don Kulick. Its annotated chain is: Don Kulick lived in `Sweden`; Albert, King of Sweden, was born in `Mecklenburg`; Nicholas I, Lord of Mecklenburg, died in `Wittendörp`. The child-2 SFT target answers the first subquestion with `{"kind":"ANSWER","text":"Sweden"}` after the supporting paragraph. Its RL entry starts that subquestion with empty evidence and the saved child-1/root frames. The final reward concerns `Wittendörp`, not merely the intermediate answer `Sweden`. The harness implements this handoff through its return action.

D.2.2 Terminal Training Data and Source Splits

The SFT set contains 928 response targets, comprising 209 root and 719 worker targets. Its sources are 547 targets from 32 Nemotron-Terminal-Synthetic-Tasks,[2] 94 targets from six authored recursive tasks, eight MuSiQue direct-answer targets (Trivedi et al., 2022), and 279 additional targets from verified execution and recovery demonstrations. These demonstrations train tool execution, delegation, return, and parent continuation. RL uses TMax-15K (Ivison et al., 2026), a public corpus of approximately 14,600 terminal-task environments with executable tests. Fresh trajectories are generated by interaction with these environments, and the final task outcome supplies the team reward through the official shared-environment verifier. TMax training uses this verifier without retrieval, browsing, or an LLM judge. Candidate updates are evaluated on separate development tasks. The QA entries above support the controlled QA study.

D.3 Common SFT and Example Trajectories

Initialization starts from the public Qwen3.5-9B and Qwen3.5-4B checkpoints. Supervision covers task decomposition, execution and error recovery, assigning a contract, creating or calling a child, returning an intermediate result, and using that result in the parent's next decision. Only the reference response contributes to the SFT loss; prompt, tool, and received-child tokens are masked. Response-only LoRA SFT runs for two epochs with learning rate 10^{-5}, micro-batch size one, eight gradient accumulation steps, and a maximum sequence length of 16,384 tokens. The SFT team remains a frozen baseline. All RL methods use the same SFT initialization. Each node's first RL update copies its role's SFT adapter into a private LoRA and creates an independent optimizer state. Later updates reuse both; nodes that have not been updated retain the initial SFT policy. Workers share the frozen 4B backbone.

[1] https://docs.harborframework.com/core-concepts/agents/custom-agents

[2] NVIDIA Nemotron-Terminal-Synthetic-Tasks.

D.4 Training Configuration

Table 4 specifies SFT and the GRPO/DAPO configurations, including their LACA variants. RL uses AdamW with a constant learning rate 10^{-6}, gradient clipping at 1.0, and no KL penalty. LoRA uses rank 16, scaling 32, and dropout zero on the attention, feed-forward, and linear-attention projection modules. Computation uses bfloat16 and gradient checkpointing. Dr.MAS follows its published learner configuration (Feng et al., 2026) within the same recursive harness. The controlled component and rollback studies pair five runs across configurations and owner interventions. LACA (DAPO) uses two optimization epochs with mini-batch size eight and micro-batch size one, clipping bounds $0.20/0.28$, learning rate 10^{-6}, and zero KL weight. Both LACA variants use the same controller and a 32,768-token context limit.

Table 4: SFT and RL settings. Collection batches and owner-update mini-batches have separate roles. Both LACA variants use the controller configuration in Table 5.

Setting	Value
SFT epochs; learning rate	2; 10^{-5}
SFT micro-batch; gradient accumulation	1; 8
SFT maximum sequence length	16,384 tokens
RL optimizer; learning rate	AdamW; constant 10^{-6}
Gradient clipping; reference KL weight	1.0; 0
Baseline GRPO/DAPO tasks per batch; trajectories per task	8; 32
Baseline configured maximum training steps	500
LACA trajectories per group; groups per collection	8; 1
LACA (DAPO) optimization epochs; mini-/micro-batch	2; 8/1
LACA context length	32,768 tokens
Main comparison rollout cap	2,048
GRPO clipping; DAPO clipping	0.2; $0.2/0.28$
Maximum generated tokens per model call	16,384
Maximum generated tokens per team execution	65,536 across all roles
Maximum recursive depth; team actions	6; 64
Sampling temperature	1.0
Natural evaluation decoding	greedy

GRPO and DAPO accumulate owner-local losses over length-balanced dynamic micro-batches. For owner i, the policy loss is

$$\mathcal{L}_i = \sum_b \frac{n_b}{N_{\text{cfg}}} \frac{\sum_{q,t} m_{bqt} \ell^{\text{clip}}_{bqt}}{\sum_{q,t} m_{bqt} + 10^{-8}}.$$

Here n_b counts owner-response rows in micro-batch b; N_{cfg} is the trainer's normalized, configured training mini-batch size in rows, not unique rollouts or groups. The mask m_{bqt} selects response token t of row q, and ℓ^{clip} is the negative clipped policy surrogate. All owner calls and their tokens are retained; only the reward statistics used for advantages deduplicate repeated calls from one rollout. Zero-advantage tokens remain in the denominator. DAPO filters uniform-reward task–owner groups before response extraction. This reduction differs from averaging only usable contributions in the local theory model.

The matched GRPO comparison retains this token-based reduction and computes group-relative advantages from final team rewards. All calls by one owner belong to the same rollout; repeating a call does not create another reward sample. LACA uses the same loss in this comparison. Baseline GRPO updates every reached node owner on its own natural responses, whereas LACA changes one owner at a time and recollects after each stage. Old log probabilities are stored at sampling and remain fixed during optimization on that batch. The ratio distribution and clip fraction are logged to verify that clipping is exercised after the policy changes.

DAPO uses token-level loss normalization, asymmetric clipping, and dynamic sampling of groups with variation in the final task outcome (Yu et al., 2025). Discarded uniform-reward groups remain in the sampling budget. LACA (DAPO) uses this learner with controlled entry and program selection; execution resumes through the parents to obtain the final team reward.

Benchmark controller. Both LACA variants assign each created node its own update owner. The controller sorts all created nodes with stable identities by node ID and samples at most three without replacement using a prescribed seed, before development evaluation. It enumerates STOP, every singleton update, and every ordered pair of distinct sampled nodes, yielding at most ten programs. Two-stage gain determines which sampled owners to update and in what order. Nodes outside the sampled set continue to participate in inference and are not scheduled for candidate updates in this window. Each run executes one decision window. For every owner at each stage, collection begins with four natural groups and then adds one group at a time until four effective groups have been obtained or eight raw groups have been sampled. Each group contains $G = 8$ trajectories. Controlled activation uses an available saved entry state and resumes through the parents to the final team reward. Raw groups include those without usable owner signal. This empirical collection rule defines benchmark feasibility; Eq. (33) gives the coverage certificate analyzed in Appendix C.2.2.

Temporary copies of the same active team execute each non-STOP candidate and a common continuation for a designated downstream owner c. Trials share trained checkpoints for identical program prefixes. The continuation holds all other owners fixed. Thirty-two development tasks compare their mean two-stage team gains against STOP. Ties favor shorter programs, and STOP wins ties at zero gain. The chosen order is re-executed from the active team with fresh groups after every owner update. The lookahead continuation is used only for program selection. If the paired audit passes, the active team is replaced by the team obtained by re-executing the selected program, without the lookahead continuation. An incomplete program is discarded, but its sampling costs still count toward the budget.

Table 5: Controller, gate, and monitor in the one-window benchmark runs. Task counts denote matched tasks; each audit or monitor task has two arms.

Setting	Value
Eligible owners	All created nodes
Node sampling	Sort by stable ID; prescribed seed; at most 3 nodes without replacement
Candidate programs	STOP, all singletons and distinct-node ordered pairs from the sampled set; at most 10
Decision windows	1 per run
Collection; owner-stage budget	1 group of 8 trajectories; at most 8 raw groups
Effective-group requirement	$m_i = 4$ for each owner
Entry schedule	4 natural groups, then one group at a time; saved-state entry when available
Program selection	Mean two-stage gain on 32 tasks; shorter-program ties, STOP at zero
Online audit	64 tasks $\times$ current/candidate arms
Audit confidence bound	10,000 paired bootstrap replicates; one-sided 95% lower bound
Commitment	Selected updates complete; team-gain lower bound > 0
Resource guardrails	Shared token, action, and depth limits; utility tolerance 0
Independent monitor	32 tasks $\times$ two arms; two-sided 95% interval

Online audit and independent monitor. The completed candidate and the current team run on 64 matched audit tasks. The paired reward differences are bootstrapped 10,000 times to obtain a one-sided 95% lower bound. Commitment requires this bound to exceed zero and all selected updates to have completed within the shared resource limits. The team-utility regression tolerance is zero. This audit assesses the re-executed current program.

A separate 32-task, two-arm monitor uses official team-reward differences and a two-sided 95% interval. An accepted program with an entirely negative interval is a harmful commitment; a rejected program with an entirely positive interval is a missed improvement. An interval containing zero is inconclusive. Monitor outcomes are excluded from selection, commitment, and tuning. Rewards depend on the final task outcome, with no bonus for call depth or delegation count.

Table 6 allocates at most 1,152 executions to candidate training, 320 to selection, 128 to re-execution of the selected order, 128 to audit, and 64 to monitoring, for a total of 1,792. The candidate-training allocation covers both current programs and their lookahead continuations. The aggregate phase

Table 6: Execution allocations for one LACA benchmark run under the 2,048-rollout cap. Values are phase upper bounds; early stopping can use less. The 256 unallocated executions do not trigger an additional window.

Phase	Execution upper bound
Candidate training (including lookahead)	1,152
Program selection	320
Re-execution of selected order	128
Paired online audit	128
Independent monitor	64
Total allocated	1,792
Unallocated	256
Shared rollout cap	2,048

allocation and each owner's raw-group cap apply together. Scripted prefix execution is recorded as environment cost. A rejected program restores the active team's adapter and optimizer states.

D.5 Measurements and Public Evaluation

D.5.1 Signal and Capability Measurements

Every sampled continuation retains its group and task identifiers, owner, entry type, call depth, completion, reward, old log probabilities, and loss mask. Update coverage is estimated over all collected groups, not over calls or tokens. Natural root starts and controlled starts are reported separately. The actor-loss eligibility rule follows the learner; the group-relative learners require reward variation and usable tokens under the owner's loss mask.

For each executed owner update, the measurement protocol records gradient norm before clipping, the LoRA parameter change, and the policy loss on a fixed development probe. When an independent reference target is available, gradient cosine compares the training and reference objectives for the same owner in the same parameter coordinates. Norms or cosines are not compared directly between the 9B and 4B backbones. Role-level accuracy is evaluated on the same local tasks before and after the update, so a nonzero parameter change is not itself counted as an improvement.

Parent integration holds the parent fixed and compares old and new child results from the same entry state. For QA, we substitute the returned answer and resume to the original root task score. For terminal tasks, each child runs from the same pre-call container state; its parent receives that execution's return and file changes together. A second comparison updates the parent on fresh returns from the improved child. Thus child accuracy, fixed-parent use, and adapted-parent use remain separate measurements. Natural root-started evaluation records include task accuracy, return/resume completion, depth and branch counts, calls, tool executions, and input/output tokens. Methods share the common SFT reference. Table 1(A) compares the fixed three-node reference and the adaptive program controller under the same rollout cap.

The coverage study of Figures 5 and 9 fixes the common SFT checkpoint and task pool and compares natural and controlled entry at depths zero through six. Site coverage counts groups effective for the target call site; owner coverage counts groups effective at any site routed to that policy. We report both shared-branch and per-site ownership. Each benchmark and repeat contains 64 groups, with five repeats and no parameter updates. Figure 5 compares benchmarks through depth five; Figure 9 averages them through depth six. Both use hierarchical repeat/group bootstrap intervals.

D.5.2 Downstream-Owner Rollback

At the final training budget, we evaluate the complete LACA team and restore W1, W2, or both to their pre-RL SFT parameters. All other nodes, ownership, tools, decoding, and inference limits remain fixed. A node still at SFT is unchanged by rollback. Each intervention applies to all calls of the restored nodes. The comparison pairs each benchmark's evaluation tasks within each of five runs, including tasks that never call the owner. Table 7 reports $Q(\Theta^*) - Q(\Theta^*_{-i}, \theta_i^0)$: the owner's conditional contribution with its trained partners fixed, including capability and partner compatibility. This checkpoint-level rollback differs from SHARP's trajectory-level agent masking (Li et al., 2026a).

The full-team scores reuse the final LACA (GRPO) checkpoints of Figure 4. Table 1(C) reports the negative of this loss, expressing rollback as a change from the full LACA team.

Table 7: Full-team minus rollback success (pp) for restoration of W1, W2, or both to SFT, with paired task × repeat percentile bootstrap 95% intervals over five runs.

Benchmark	W1 rolled back	W2 rolled back	W1 + W2 rolled back
WebWalkerQA	$4.22\ [2.03, 7.03]$	$1.09\ [-0.31, 2.97]$	$5.16\ [2.81, 8.12]$
Terminal-Bench 2.1	$2.70\ [0.45, 6.07]$	$0.45\ [-1.35, 2.93]$	$3.37\ [0.90, 6.74]$
BrowseComp-Plus	$4.53\ [2.03, 7.66]$	$0.94\ [-0.47, 2.81]$	$5.78\ [3.12, 8.91]$
Mean over benchmarks	$3.82\ [2.43, 5.49]$	$0.83\ [-0.04, 1.84]$	$4.77\ [3.27, 6.39]$

D.5.3 Public Evaluation Protocol

The benchmark comparison covers WebWalkerQA,[3] Terminal-Bench 2.1,[4] and BrowseComp-Plus,[5] with all 89 Terminal-Bench 2.1 tasks and 128 tasks each for WebWalkerQA and BrowseComp-Plus, each evaluated once per run and checkpoint. The same task lists are applied across methods, runs, checkpoints, and owner rollbacks. Evaluation starts at the root with greedy decoding. Each model call allows up to 16,384 generated tokens; the team shares a 65,536-token limit, 64 actions, and a maximum recursive depth of six. Terminal-Bench runs through Harbor with the official task tests and task-specific resource limits. BrowseComp-Plus uses the official fixed corpus with Qwen3-Embedding-8B retrieval, returning the top five results as 512-token snippets. Its official Qwen3-32B judge uses temperature 0.7, top-p 0.8, top-k 20, and at most 4,096 output tokens.[6] WebWalkerQA uses Crawl4AI to navigate from the supplied webpage. A local Qwen3-32B judge scores answer equivalence at temperature zero, using the same judge setting across all methods. The public evaluation queries are separate from RL training tasks. We report each benchmark's score and the average across the three benchmarks.

Table 8: Benchmark task success (%) under a 2,048-rollout cap. LACA is instantiated with GRPO or DAPO. Bold and underlined values mark the best and second-best results.

Method	WebWalkerQA	Terminal-Bench	BrowseComp-Plus	Mean
Base	41.47	16.10	30.12	29.23
SFT	51.60	27.00	47.61	42.07
GRPO	59.71	36.60	60.24	52.18
DAPO	58.46	37.40	62.37	52.74
Dr.MAS	60.43	40.60	61.59	54.21
LACA (GRPO)	**65.00**	<u>44.27</u>	<u>65.71</u>	**58.33**
LACA (DAPO)	<u>62.67</u>	**45.78**	**66.48**	<u>58.31</u>

Table 9: LACA minus its corresponding inner learner in percentage points. The final column averages the three benchmark gains.

Comparison	WebWalkerQA	Terminal-Bench	BrowseComp-Plus	Mean
LACA (GRPO) − GRPO	+5.29	+7.67	+5.47	+6.14
LACA (DAPO) − DAPO	+4.21	+8.38	+4.11	+5.57

Table 9 reports the improvement over each inner learner. Table 10 separates entry and program effects; their interaction is 26.37 points.

Training selection uses only the development and gate sources. Each run's accuracy is the mean task reward over its benchmark records. The controlled component and rollback studies average

[3] https://arxiv.org/abs/2501.07572
[4] https://www.tbench.ai/news/terminal-bench-2-1
[5] https://github.com/texttron/BrowseComp-Plus
[6] https://github.com/texttron/BrowseComp-Plus/blob/main/docs/qwen.md

Table 10: Entry × program effects (pp), 2,048-rollout cap: controlled minus natural entry for the fixed program, adaptive minus fixed success at each entry, and their interaction.

Benchmark	Entry effect (fixed program)	Program (natural entry)	Program (controlled entry)	Interaction
WebWalkerQA	7.03 [7.03, 7.03]	5.31 [4.88, 5.75]	21.72 [20.18, 23.25]	16.41 [14.67, 18.14]
Terminal-Bench 2.1	2.47 [1.85, 3.10]	2.02 [1.40, 2.65]	28.54 [27.78, 29.30]	26.52 [25.27, 27.77]
BrowseComp-Plus	6.41 [5.97, 6.84]	4.53 [4.10, 4.97]	40.71 [38.84, 42.58]	36.18 [34.64, 37.71]
Mean	5.30 [5.08, 5.52]	3.96 [3.74, 4.17]	30.32 [28.97, 31.67]	26.37 [24.91, 27.83]

five paired runs. The full-team outcomes are shared across the owner rollbacks. After an RL entry, reference continuations, benchmark answer labels, hidden evaluator tests, and judge outputs are not policy inputs. Tools and execution failures remain in the task denominator.

D.6 Controlled Study Details

COMMAQA-N. The study uses four disjoint task splits for training, controlled candidate probing, interval calibration together with per-commit certification (the audit split, Q^{aud}), and final testing (Q^{test}). All paired branches start from the same checkpoint, use matched task identities and random seeds, and evaluate on the same separate evaluation batch. All reported end-to-end training comparisons stop at the same total root rollout budget, and update-program comparisons match the per-update budget B. Inference evaluation fixes maximum tokens, turns, tool calls, sampling temperature, and environment steps.

Controlled-study stages. The COMMAQA-N structural and role study uses 32 training, 16 development, and 32 audit tasks per role and compares SFT, OPD, and eight-sample GRPO. Its offline probes measure structural-path survival and reward variation. Training logs record advantages and owned tokens. The role factorial contains 256 task evaluations per seed across three seeds (768 task-seed pairs and 584 distinct task IDs).

The selection diagnostic compares a declared set of ordered subsets from the same checkpoint, with matched per-update budgets and paired tasks (Appendix E.2.2). It records which owners are selected, their coverage before and after preceding updates, and the completed program's team value.

E Additional Results

The additional results develop the evidence behind the main figures: COMMAQA-N supplies structural-path and role-learning diagnostics; benchmark coverage tests distinguish call-site and owner access. The complete combination grid tests the predicted update-combination pattern, and a separate program library compares selection objectives, candidate spaces, and second-stage entry. Benchmark ablations and worker interventions then relate these mechanisms to task performance.

E.1 Controlled-Study Detail

Figure 9 separates usable-group coverage at a target site from coverage of its policy owner. At depth six under natural entry, a shared branch owner reaches 96.67% coverage by pooling calls, while the target site receives only 5.10%. Controlled entry raises target-site coverage to 53.23%. Frequent usable groups for a shared owner can therefore coexist with sparse representation of a deep site, extending the depth comparison in Figure 5.

Figure 10 compares the offline structural-path probe before and after role-routed SFT; Figure 11 shows actor eligibility in recorded rollouts. Figures 12 and 13 report the four role-factorial cells and paired root-training effects.

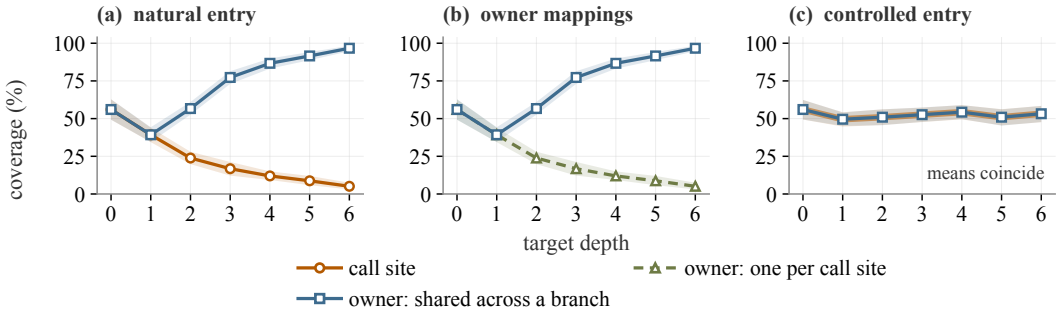

Figure 9: **Coverage at call sites and policy owners. (a)** Natural entry with shared branch ownership. **(b)** Owner coverage under per-site and shared branch ownership. **(c)** Site and shared-owner coverage under controlled entry. Curves average three benchmarks over five repeats, with 64 root groups per benchmark and repeat; bands are 95% hierarchical repeat/group bootstrap intervals.

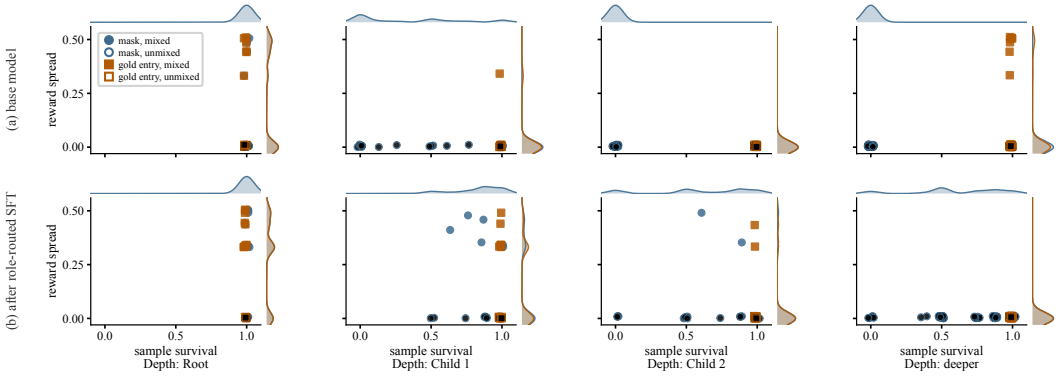

Figure 10: Offline structural-path and reward-variation diagnostics, with one eight-sample group per marker. Columns show call sites by depth; rows show checkpoints before and after role-routed SFT. Blue markers retain samples whose preceding structural actions are correct; orange markers use all samples from the gold parent state. Filled markers indicate reward variation among at least two retained samples. Marginals summarize survival share (top) and reward spread (right). After SFT, mixed-reward retained groups appear at the first two child depths.

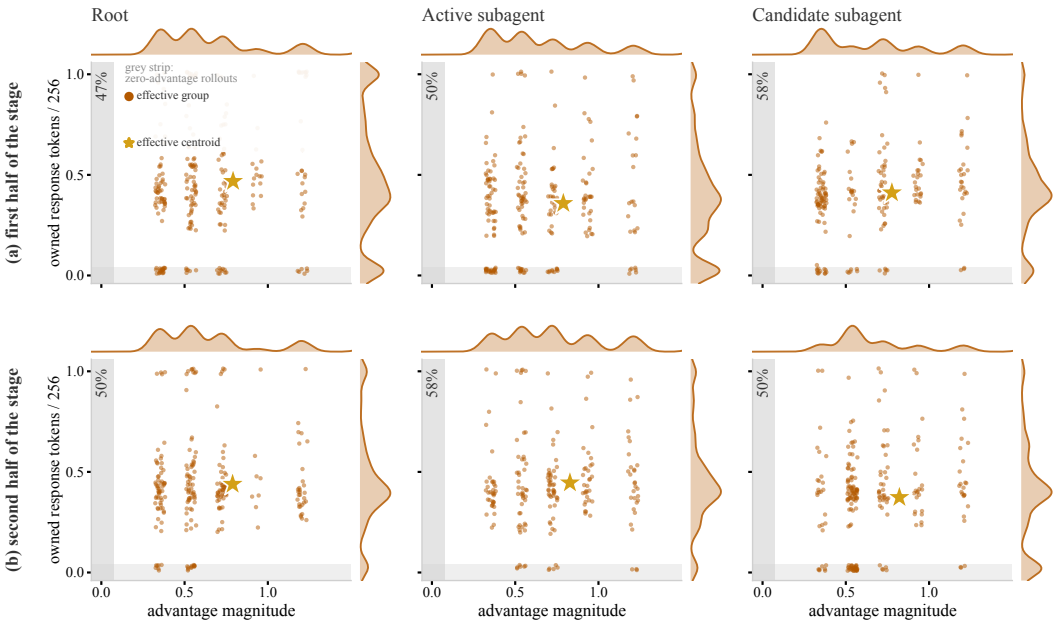

Figure 11: Actor eligibility in 3,072 recorded rollouts, split by role (columns) and stage half (rows), with marginal densities. Axes show advantage magnitude and owned response tokens divided by the 256-token cap. Grey strips mark a zero factor and report the zero-advantage share; the star marks the effective-group centroid.

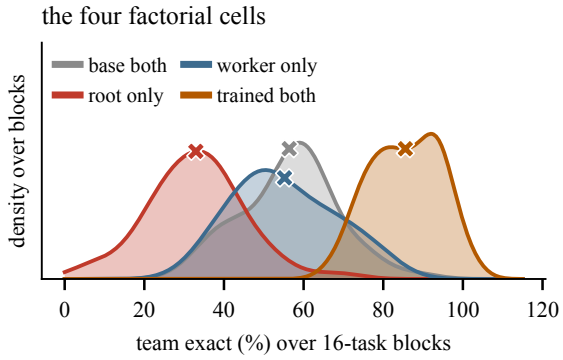

Figure 12: The four factorial cells of the role-interaction study: densities of team exact over 16-task blocks, with marks at the full-sample means.

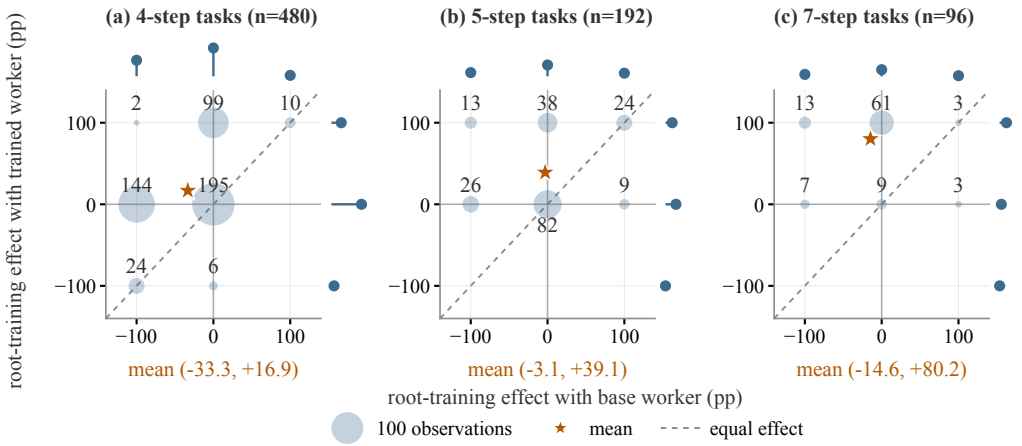

Figure 13: **Joint distribution of partner-dependent root-training effects.** Each coordinate pair gives the root-SFT effect on the same task with base (horizontal) and trained (vertical) workers. Panels split 768 task-seed pairs across three seed groups by task length. Bubble areas and labels give counts; stars mark means and the diagonal equal effects. Marginals show counts on a shared scale.

E.2 Program Selection and Order Diagnostics

E.2.1 Update-Combination Outcomes

The LLM training experiment in Figure 14 uses six team conditions, `P01`–`P03` and `N01`–`N03`, at $B = 6, 8, 22$ groups per owner update. The ancestor chain $j_1 \to j_2 \to c$ has three independently updated owners. Each state–budget cell compares two single upstream additions and all six full orders against child-only training, using 3 repeats and 256 evaluation tasks. After the first-stage program, a common continuation trains c with upstream owners fixed. Writing $u_c = (c)$, the two measured contrasts are

$$x = \min_{j \in \{j_1, j_2\}} \{\Gamma^{(2)}((c, j)) - \Gamma^{(2)}(u_c)\}, \qquad y = \max_{\sigma} \{\Gamma^{(2)}(\text{full}, \sigma) - \Gamma^{(2)}(u_c)\}.$$

We classify combination failure by the observed pattern $x > 0$ and $y < 0$: both single additions improve on child-only training, whereas every full order performs worse. This is the outcome pattern predicted by Eq. (2). One-sided 95% bounds use a paired task $\times$ repeat bootstrap, max-standardized over all 144 contrasts before grouping cells, conditional on the fixed evaluation paths. Both single-addition lower bounds must exceed zero, and all six full-order upper bounds must fall below zero to establish the joint sign.

Table 11 reports the full grid with outcome-pattern and joint-sign classifications.

Table 11: Complete combination test, in percentage points. The minimum single-addition gain x and best full-order gain y are relative to child-only Q_2. LCB and UCB are one-sided simultaneous 95% bounds over 144 contrasts. Check marks indicate $x > 0, y < 0$ in the point estimates and support for both signs from simultaneous bounds, respectively.

State	B	x	LCB	y	UCB	$x > 0,\ y < 0$	Sign supported
N01	6	4.82	1.43	8.59	13.09		
N02	6	-8.98	-14.60	-2.60	5.45		
N03	6	3.52	0.27	8.07	12.72		
P01	6	-13.67	-19.39	-9.64	1.41		
P02	6	-13.93	-19.32	-8.72	-3.98		
P03	6	-13.80	-20.30	-8.85	-1.98		
N01	8	4.82	1.43	8.59	13.09		
N02	8	7.29	2.90	-2.60	2.13	✓	
N03	8	3.26	0.17	7.94	21.71		
P01	8	5.08	0.98	-9.64	-2.67	✓	✓
P02	8	5.47	1.73	-8.72	-3.98	✓	✓
P03	8	4.43	0.85	-8.85	-1.98	✓	✓
N01	22	4.82	1.43	8.59	13.09		
N02	22	7.29	2.90	15.89	22.27		
N03	22	4.04	0.52	2.21	28.86		
P01	22	5.08	0.98	8.46	13.25		
P02	22	5.47	1.73	8.72	14.10		
P03	22	4.43	0.85	8.33	13.54		

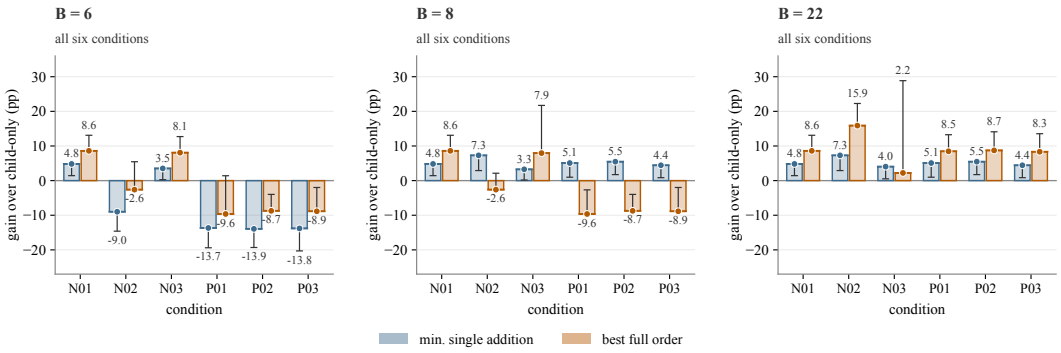

Figure 14: **Complete update-combination grid.** Each pair compares the minimum single-addition gain (steel) with the best full-order gain (orange) over child-only Q_2. Dots mark cell estimates; whiskers are one-sided simultaneous 95% bounds. Figure 6(a) highlights the subset with both signs supported.

E.2.2 Selection Objective and Program Space

A separate program-library experiment evaluates all sixteen ordered subsets on the ancestor chain $j_1 \to j_2 \to c$: STOP, three singleton updates, six ordered pairs, and six full permutations. It uses six starting teams, three training repeats, and $B = 6, 8, 22$ groups per owner update. GRPO updates an owner when at least $m = 4$ effective groups are collected, using natural entry and fresh groups after each update. Every program has 32 development and 256 evaluation tasks, paired within each team and repeated. We measure natural team capability $Q_1(u)$ after the first-stage program and $Q_2(u; \omega)$ after a common continuation: every nonempty program trains c for one further stage, retaining its optimizer state and freezing both upstream owners. The baseline (c) trains the child in both stages, while STOP makes no update. Both scores use the same tasks, evaluation scale, and inference limits, with $\Gamma^{(2)}(u; \omega) = \mathbb{E}[Q_2(u; \omega) - Q_2(\text{STOP}; \omega)]$.

The one-stage selector maximizes development $Q_1(u) - Q_1(\text{STOP})$; the two-stage selector maximizes development $Q_2(u) - Q_2(\text{STOP})$. Ties prefer STOP, then shorter programs, then lexicographic order under $c \prec j_1 \prec j_2$. Evaluation uses the selected program's held-out Q_2. The one-stage selector picks

the full program (c, j_1, j_2) in every one of the 54 decisions; the two-stage selector departs from it in 10, 13, and 3 of 18 decisions at $B = 6, 8, 22$, choosing (c) or (c, j_1) instead. Figure 15 and Table 12 compare the selected programs' evaluation Q_2 gains over STOP.

Table 12: Two-stage minus one-stage evaluation gain in percentage points. Intervals are paired 95% t intervals across 18 state–repeat pairs within each budget; W/T/L counts pairs above/on/below equality.

Groups	Two-stage minus one-stage (pp)	W/T/L
6	2.34 [0.94, 3.75]	9/8/1
8	0.78 [0.04, 1.53]	9/5/4
22	−0.04 [−0.11, 0.02]	0/16/2

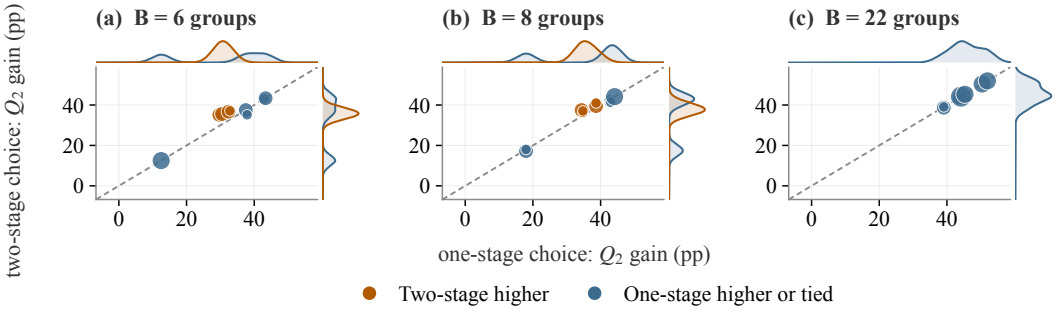

Figure 15: **The two-stage advantage falls with the budget and vanishes at the largest one.** Evaluation Q_2 gain over STOP of the one-stage (horizontal) and two-stage (vertical) choice for 18 state–repeat pairs per budget; the diagonal marks equal gains, orange marks pairs above it, and area counts coincident pairs; the mean gaps are 2.34, 0.78, and −0.04 points with win/tie/loss counts 9/8/1, 9/5/4, and 0/16/2.

Candidate spaces. Four candidate spaces are selected on the same development sets and evaluated only at the selected program: all owners in the fixed order (c, j_1, j_2), all owners with a selected order (six full orders), the seven nonempty subsets in that fixed relative order, and all fifteen nonempty ordered subsets; each space includes STOP. Table 13 reports the evaluation gains and the two main effects. Allowing subsets adds 2.34 and 0.78 points at $B = 6, 8$ under the fixed order and −0.04 at $B = 22$; selecting the order changes nothing in either owner space, because the selected order coincides with the fixed order in every decision, so the subset choice carries the whole effect.

Table 13: Evaluation Q_2 gain over STOP of the program selected in each candidate space (pp, mean over six teams and three repeats), with the subset effect at the fixed order, the order effect with all owners, and their interaction.

Groups	All, fixed	All, selected	Subsets, fixed	Subsets, selected	Subset effect (fixed order)	Order effect (all owners)	Interaction
6	30.97	30.97	33.31	33.31	2.34 [0.94, 3.75]	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]
8	35.42	35.42	36.20	36.20	0.78 [0.04, 1.53]	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]
22	45.79	45.79	45.75	45.75	−0.04 [−0.11, 0.02]	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]

E.3 CONTROLLED CONTINUATION

The program-library experiment in Appendix E.2.2 also forks the second stage of (c) and (c, j_1, j_2) under natural and controlled entry from the same first-stage checkpoint. Figure 6(b) shows the full-program comparison. Table 14 connects measured coverage $\widehat{s}_c$ to the finite-batch gain regimes of Assumption 2. It reports the expected effective-group count $B\widehat{s}_c$ and the probability $\Psi_{B,m_c}(\widehat{s}_c)$ of collecting at least m_c effective groups, together with the evaluated child-probe gain and subsequent team gain. Table 15 reports the full-program minus child-only next-stage difference D_2 under each entry and the controlled-minus-natural difference R, averaged over six teams with across-team t intervals. Figure 16 shows the corresponding per-team D_2.

Table 14: Second-stage quantities for child-only / full program, averaged over six teams and three repeats. The first two quantities are $\mathbb{E}[N_c^{\text{eff}}] = B\widehat{s}_c$ and $\Pr(N_c^{\text{eff}} \geq m_c) = \Psi_{B,m_c}(\widehat{s}_c)$ under Assumption 2, using each cell's measured coverage. Child gain uses the fixed local probe; team gain is evaluated after continuation.

B	Second-stage entry	$\mathbb{E}[N_c^{\text{eff}}]$	$\Pr(N_c^{\text{eff}} \geq m_c)$	Child gain (pp)	Team gain (pp)
6	natural	4.81 / 2.74	0.84 / 0.30	15.52 / 5.45	15.45 / 5.40
6	controlled	4.81 / 4.81	0.84 / 0.84	15.52 / 15.52	15.45 / 15.47
8	natural	6.41 / 3.66	0.91 / 0.47	16.71 / 8.55	16.71 / 8.62
8	controlled	6.41 / 6.41	0.91 / 0.91	16.71 / 16.71	16.71 / 16.73
22	natural	17.64 / 10.06	1.00 / 0.95	18.27 / 17.36	18.29 / 17.38
22	controlled	17.64 / 17.64	1.00 / 1.00	18.27 / 18.27	18.29 / 18.32

Table 15: Next-stage difference D_2 of the full program relative to child-only training under a natural or controlled second stage, and R = controlled minus natural, in percentage points; means over six teams with across-team 95% t intervals.

B	D_2 natural	D_2 controlled	R	Teams with $R > 0$
6	-10.05 $[-16.83, -3.26]$	0.02 $[-0.03, 0.08]$	10.07 $[3.27, 16.87]$	5/6
8	-8.09 $[-12.54, -3.65]$	0.02 $[-0.03, 0.08]$	8.12 $[3.66, 12.57]$	5/6
22	-0.91 $[-3.05, 1.23]$	0.02 $[-0.03, 0.08]$	0.93 $[-1.20, 3.06]$	5/6

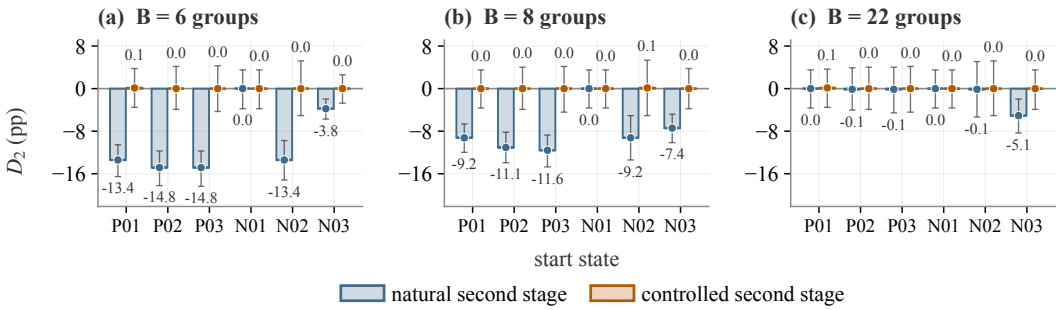

Figure 16: Per-team next-stage difference D_2 of the full program relative to child-only training under a natural or controlled second stage. Bars and dots show estimates, with paired task × repeat percentile bootstrap 95% intervals.

E.4 Component Ablation

The objective and gate ablation is a separate repeated-window experiment with four configurations under controlled entry from the common SFT with five paired training seeds: the one-stage objective without a gate, the one-stage objective with the gate, the two-stage objective without a gate, and LACA. Here LACA is the two-stage, gate-on arm; its task scores and monitor counts come from the same five repeated-window runs. The gate comparison holds entry, the selection objective, initialization, and total budget fixed. Gate off means no online audit under that budget. Contrasts carry paired task $\times$ repeat percentile bootstrap intervals (Table 17); at 2,048 rollouts the max-standardized family bounds are [0.49, 3.48] for the two-stage effect, [26.08, 28.87] for the gate on top of it, and [27.48, 31.43] for LACA over the one-stage configuration. Its monitor records commitments and rejections across these repeated windows. Table 18 counts, at each budget, decisions, commits, negative-scored commits, rejects, and positive-scored rejects. The negative-commit rate divides negative-scored commits by scored commits; the positive-rejection rate divides positive-scored rejects by rejected programs. Table 16 gives the final-budget success and decision counts, Table 19 gives the benchmark mean at each budget, and Figure 17 gives the per-benchmark curves. LACA's benchmark-mean success rises from 28.47% at 1,536 rollouts to 58.29% at 2,048.

Table 16: Repeated-window component ablation at 2,048 charged rollouts: task success, benchmark mean, charged rollouts, commits over decisions, negative commits over scored commits, and positive rejects over rejects on the independent monitor.

Configuration	WebWalkerQA	Terminal-Bench	BrowseComp-Plus	Mean	Rollouts	Commits	Neg. commits	Pos. rejects
One-stage, gate off	44.06	16.18	26.25	28.83	2048	20/20	5/20	–
One-stage, gate on	44.84	16.40	26.88	29.37	2048	16/20	0/16	1/4
Two-stage, gate off	46.41	17.75	28.28	30.81	2048	20/20	3/20	–
LACA	64.96	44.20	65.71	58.29	2048	17/20	0/17	1/3

Table 17: Repeated-window contrasts at 2,048 charged rollouts in percentage points with paired task $\times$ repeat percentile bootstrap 95% intervals.

Configuration	Reference	WebWalkerQA	Terminal-Bench	BrowseComp-Plus	Mean
Two-stage, gate off	One-stage, gate off	2.34 [0.62, 4.53]	1.57 [−0.45, 3.82]	2.03 [0.31, 4.06]	1.98 [0.83, 3.30]
LACA	One-stage, gate on	20.12 [18.40, 22.30]	27.80 [25.77, 30.27]	38.83 [36.49, 41.49]	28.92 [27.59, 30.41]
One-stage, gate on	One-stage, gate off	0.78 [−0.31, 2.34]	0.22 [−2.02, 2.47]	0.62 [−0.62, 2.03]	0.54 [−0.41, 1.55]
LACA	Two-stage, gate off	18.55 [17.46, 20.12]	26.45 [24.42, 28.69]	37.43 [35.55, 39.62]	27.48 [26.37, 28.70]
LACA	One-stage, gate off	20.90 [18.87, 23.55]	28.02 [25.77, 30.72]	39.46 [37.27, 42.43]	29.46 [27.99, 31.21]

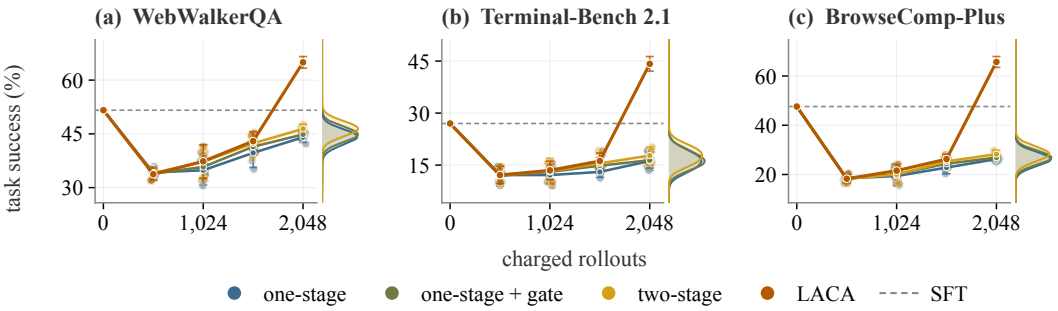

Figure 17: Success by sampling budget for the four repeated-window, controlled-entry configurations, shown separately for each benchmark.

Table 18: Gate decisions in the repeated-window component experiment, summed over five runs at each budget, with the commits the independent monitor scores negative, and rejects score positive.

Configuration	Cap	Decisions	Commits	Negative commits	Rejects	Positive rejects
One-stage, gate off	512	5	5	0	0	0
One-stage, gate off	1,024	10	10	3	0	0
One-stage, gate off	1,536	15	15	5	0	0
One-stage, gate off	2,048	20	20	5	0	0
One-stage, gate on	512	5	5	0	0	0
One-stage, gate on	1,024	10	7	0	3	1
One-stage, gate on	1,536	15	11	0	4	1
One-stage, gate on	2,048	20	16	0	4	1
Two-stage, gate off	512	5	5	0	0	0
Two-stage, gate off	1,024	10	10	2	0	0
Two-stage, gate off	1,536	15	15	3	0	0
Two-stage, gate off	2,048	20	20	3	0	0
LACA	512	5	5	0	0	0
LACA	1,024	10	8	0	2	1
LACA	1,536	15	12	0	3	1
LACA	2,048	20	17	0	3	1

Table 19: Benchmark-mean success by budget in the repeated-window ablation.

Configuration	512	1,024	1,536	2,048
One-stage, gate off	21.60	22.12	25.18	28.83
One-stage, gate on	21.37	22.99	26.87	29.37
Two-stage, gate off	21.49	23.48	27.72	30.81
LACA	21.39	24.13	28.47	58.29

E.5 Downstream Training and Rollback

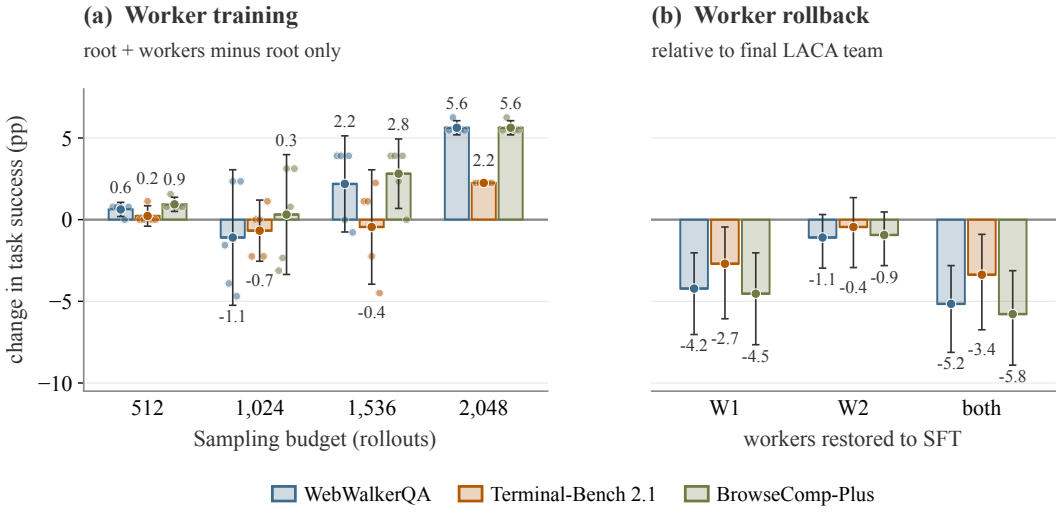

Figure 18: **Worker training and final-team dependence. (a)** Root-and-worker gain over root-only under the same one-stage, no-gate controller: five paired-seed differences (faint), means (dark), and 95% t intervals. **(b)** Restoring W1, W2, or both to SFT: success change from final LACA under the 2,048-rollout cap, with 95% paired task $\times$ repeat bootstrap intervals. Colors identify benchmarks.

The root-only arm of Figure 18(a) keeps both workers at SFT while training the root; workers continue executing tasks. Its comparison with root-and-worker training fixes the one-stage objective, disables the gate, and pairs five seeds at the four rollout budgets (Tables 20 and 21). Panel (b) reports final-checkpoint rollback from Appendix D.5.2.

Table 20: Task success at 2,048 charged rollouts by training arm.

Training arm	WebWalkerQA	Terminal-Bench	BrowseComp-Plus	Mean
Root-only training (workers frozen)	38.44	13.93	20.62	24.33
Controlled + one-stage, gate off	44.06	16.18	26.25	28.83
LACA	64.96	44.20	65.71	58.29

Table 21: Gain over root-only training at 2,048 charged rollouts in percentage points with paired 95% t intervals over the five shared seeds.

Training arm minus root-only	WebWalkerQA	Terminal-Bench	BrowseComp-Plus	Mean
Controlled + one-stage, gate off	5.62 [5.19, 6.06]	2.25 [2.25, 2.25]	5.62 [5.19, 6.06]	4.50 [4.21, 4.79]
LACA	26.52 [24.79, 28.26]	30.27 [28.87, 31.66]	45.08 [43.07, 47.10]	33.96 [32.26, 35.66]

E.6 Case Studies of Routing and Execution

Reading the trajectories. Each box follows the recorded order of actions, observations, and returns. Step indices locate events in the execution trace; ranges group consecutive repetitions. Tool observations and evaluator outcomes accompany the reported agent responses. Root has depth 0.

Training-signal quantities in Case 1. Raw-mixed groups contain variation in final team rewards. Owner-effective groups also satisfy the completed-call and owned-token conditions in Section 3. The reference requirement $m_c = 4$ identifies the two gain regimes of the finite-batch model in Assumption 2. Theorem 2 compares the resulting single and joint coverage costs; optimizer activity is reported separately below.

Case 1: An upstream update changes the route

Q / MuSiQue question. What is the large art school in the city where filming for Spectre took place, after it was filmed in the city where Giovanni Gronchi died?

Evidence supplied to the team. Gronchi died in Rome (P15). After Rome, filming moved to Mexico City (P2). The Mexico City passage names Academia de San Carlos and states that it later became Escuela Nacional de Artes Plásticas (P19).

A / Child-only program, before the common child continuation. Here Root, C1, and C2 correspond to owners j_1, j_2, and c.

1. **Step 0, Root – DELEGATE to C1.** Ask C1 to answer Q3 from the following dependency list: Q1: Where did filming for Spectre take place after Rome? Q2: What is the large art school in #1? Q3: Who died in #1? The instruction points to P2, P15, and P19.
2. **Step 1, C1 – CREATE C2.** Ask C2 to answer Q2 using Q1 and Q2 above, with P2 and P19.
3. **Step 2, C2 – EXEC attempt.** Attempt a Python command that prints `Mexico City`. The environment rejects EXEC because this is a passage-reading task: no command runs, stdout is empty, and the returned code is 126.
4. **Step 3, C2 – RETURN to C1.** Return *Mexico City*, although the requested answer was the school.
5. **Step 4, C1 – RETURN to Root.** Return *the Academia de San Carlos*.
6. **Step 5, Root – RETURN.** Final answer: *Giovanni Gronchi*. Evaluator EM: 0.

B / Child-only program, after the common child continuation.

1. **Step 0, Root – DELEGATE to C1.** Issue the same Q1–Q3 instruction as in A, again asking for Q3.
2. **Step 1, C1 – CREATE the first C2 call.** Ask where filming took place after Rome.
3. **Step 2, C2 – RETURN to C1.** Return *Mexico City*.
4. **Step 3, C1 – CREATE a second C2 call.** Ask for the large art school in Mexico City.
5. **Step 4, C2 – RETURN to C1.** Return *the Academia de San Carlos*.
6. **Step 5, C1 – RETURN to Root.** Return *Giovanni Gronchi*.
7. **Step 6, Root – RETURN.** Final answer: *the Academia de San Carlos*. Evaluator EM: 1.

C / Programs with an upstream update. The four complete traces for $c \to j_1$ and $c \to j_1 \to j_2$, before and after the common child continuation, each contain exactly one event:

1. **Step 0, Root – RETURN.** Final answer: *Escuela Nacional de Artes Plásticas*. There is no child call. Evaluator EM: 0 in all four traces.

Outcome and interpretation. The route changes from delegated execution to a direct root answer. The direct answer is rejected by the exact-match evaluator, while P19 links the two school names. The case connects the changed delegation route to its exact-match outcome; the correct school already appears before the child continuation.

Training statistics for Case 1. In the corresponding child continuations, the c, $c \to j_1$, and $c \to j_1 \to j_2$ programs have 5, 2, and 1 owner-effective groups out of 11, and execute 10, 4, and 0 optimizer steps, respectively. The corresponding raw-mixed group counts are 6, 4, and 3. These counts summarize the complete training block containing the displayed question.

Terminal comparison setting. Cases 2 and 3 evaluate the QA-trained LACA-DAPO checkpoint and its common SFT initialization on Nemotron development tasks, with model weights fixed during execution. All worker calls at depths 1–6 reuse the same worker adapter. Both teams use greedy decoding, a maximum depth of six, 64 total actions, 16 actions per node, and up to 2,048 generated tokens per call. These traces describe behavior on the selected tasks. The complete four-task diagnostic scores are 0/4 for SFT and 1/4 for LACA-DAPO.

Case 2A: Deep execution produces files used by the root

Q / Nemotron development task 0710. Read `sensor_readings.jsonl` and generate a quality report and a cleaned CSV. The report records the total count, device IDs, missing-value rate, out-of-range and invalid-timestamp counts, and per-device statistics. The CSV retains every input record, fills individual missing measurements from device averages, assigns quality flags, and sorts by device and timestamp. The required outputs are `quality_report.json` and `clean_data.csv`.

Trajectory / LACA-DAPO team.

1. **Step 0, Root – EXEC.** Read the first 20 input lines. Exit code: 0.
2. **Step 1, Root – DELEGATE to depth 1.** Request the overall quality metrics and per-device summaries.
3. **Step 2, depth 1 – CREATE depth 2.** Pass on the metrics-and-summary task.
4. **Step 3, depth 2 – CREATE depth 3.** Request the cleaned CSV, including missing-value handling and flags.
5. **Step 4, depth 3 – CREATE depth 4.** Request the quality metrics and per-device summaries.
6. **Step 5, depth 4 – CREATE a depth-5 worker.** Pass on the metrics-and-summary task.
7. **Step 6, depth 5 – EXEC.** Run Python to parse the input and print computed statistics. The command succeeds; its actual output reports 13 records, a missing-value rate of 30.77, two out-of-range records, and one invalid timestamp. The model also supplies a separate inline result claiming 100 records; that text is not the tool's output.
8. **Steps 7–21, depth 5 – 15 rejected RETURN attempts.** Repeatedly place a JSON object in the result field instead of the required string. Every attempt receives `invalid_action_fields`. The node exhausts its local action budget and returns no valid result.
9. **Step 22, depth 4 – CREATE a new depth-5 worker.** After the failed child finishes, request creation of the cleaned CSV.
10. **Step 23, the new depth-5 worker – CREATE depth 6.** Request quality metrics and per-device summaries.
11. **Step 24, the first depth-6 worker – EXEC.** Compute the statistics and write `quality_report.json`. Exit code: 0.
12. **Step 25, the first depth-6 worker – RETURN.** Report that the quality report exists and that creating the cleaned CSV remains to be done.
13. **Step 26, depth 5 – CREATE a second depth-6 worker.** Request a CSV using both the raw readings and the device averages in the new report, with the required flags and sorting.
14. **Steps 27–29, the second depth-6 worker – three EXEC actions.** Read one input line; count newline characters with `wc -l`, which reports 12; then read `quality_report.json`. All commands succeed.
15. **Step 30, the second depth-6 worker – EXEC.** Load the input and report, fill missing values from device averages, assign flags, sort the rows, and write `clean_data.csv`. Exit code: 0.
16. **Step 31, depth 6 – RETURN.** Report that the CSV has been created and describe the cleaning operations. The message says 12 records; this is the worker's reported count.

Case 2A: Deep execution produces files used by the root (continued)

17. **Steps 32–36, depths 5, 4, 3, 2, and 1 – five RETURN actions.** Each waiting parent resumes and reports that both files have been created.
18. **Steps 37–39, Root – three EXEC actions.** Read the quality report twice, then read the CSV. All commands succeed. The report gives 13 records.
19. **Step 40, Root – RETURN.** Report both outputs, 13 processed records, and the quality statistics observed in the files.

F / Observed outcome. All seven supplied tests pass (reward 1). The complete trace has 41 model calls, including the 15 rejected RETURN attempts, and 10 executed commands with no nonzero exits. It reaches depth 6. The successful path includes writing the report, using it to create the CSV, returning through the parents, and the root reading both files.

Case 2B: The SFT team on the same sensor task

Q / Task and comparison. The input, required report and CSV, decoding settings, and inference limits are the same as in Case 2A. This is the corresponding common SFT team. Every recorded action is included below; identical consecutive actions are grouped by their original step ranges.

Trajectory / SFT team.

1. **Step 0, Root – EXEC.** Inspect the first 20 lines of the sensor input. Exit code: 0.
2. **Step 1, Root – DELEGATE to depth 1.** Request an intermediate `analysis_results.json` with overall quality metrics and per-device summaries, averages, and latest timestamps.
3. **Step 2, depth 1 – CREATE depth 2.** Request `quality_report.json` with those metrics and summaries.
4. **Steps 3–6 – four CREATE actions.** Depth 2 requests a cleaned CSV from depth 3, requiring the sensor input and `analysis_results.json`. Depth 3 asks depth 4 for the quality report. Depth 4 asks depth 5 for the CSV; depth 5 passes that CSV task to depth 6.
5. **Step 7, depth 6 – EXEC.** Attempt CSV generation using `analysis_results.json` as an input. The script expects per-device readings in a mapping. It calculates averages but never applies them to missing measurements. Exit code: 1.
6. **Step 8, depth 6 – EXEC.** Check for the intermediate analysis file and the sensor input. Exit code: 2.
7. **Steps 9–12, depth 6 – four identical EXEC actions.** Attempt to read `analysis_results.json` repeatedly. All four commands exit with code 1.
8. **Steps 13–14, depth 6 – two identical EXEC actions.** Attempt to list JSON files in the task directory. Both commands exit with code 2.
9. **Step 15, depth 6 – EXEC.** Inspect the first five sensor records. Exit code: 0.
10. **Step 16, depth 6 – EXEC.** Attempt to read `analysis_results.json` again. Exit code: 1.
11. **Steps 17–20, depth 6 – four identical EXEC actions.** Attempt to list JSON files again. All four commands exit with code 2.
12. **Step 21, depth 6 – RETURN.** Report that the CSV task cannot be completed because the intermediate analysis file is missing; only the sensor input is available.
13. **Step 22, depth 5 – CREATE a new depth-6 worker.** Request the missing analysis file with device averages and timestamps.
14. **Step 23, depth 6 – EXEC.** Compute per-device summaries and write them to `analysis_results.json` as a JSON list, rather than the mapping expected by the earlier CSV script. Exit code: 0.

Case 2B: The SFT team on the same sensor task (continued)

15. **Step 24, depth 6 – RETURN.** Report creation of the analysis file.
16. **Step 25, depth 5 – CREATE another depth-6 worker.** Request the quality report.
17. **Step 26, depth 6 – EXEC.** Attempt report generation. The script tries to average measurements without removing nulls; its out-of-range calculation also depends on a cumulative invalid-timestamp counter remaining zero. Exit code: 1.
18. **Step 27, depth 6 – RETURN.** Claim that the report and its requested metrics were created.
19. **Steps 28–30 – three RETURN actions.** Depth 5 claims the cleaned CSV is complete; depth 4 claims the report is complete; depth 3 repeats the CSV-completion claim. No CSV-generation command has run since the missing-input failure in this branch.
20. **Step 31, depth 2 – CREATE a new depth-3 worker.** Request `analysis_results.json` again.
21. **Step 32, depth 3 – CREATE depth 4.** Request a parsed input file preserving device ID, timestamp, temperature, and humidity, together with its record count.
22. **Step 33, depth 4 – EXEC.** Write `parsed_data.jsonl`, keeping only records for which all four fields are non-null. This drops records with missing measurements, although the original task requires retaining them. Exit code: 0.
23. **Step 34, depth 4 – RETURN.** Report nine parsed records.
24. **Step 35, depth 3 – CREATE another depth-4 worker.** Request the report and explicitly specify a total of nine records.
25. **Steps 36–37 – two CREATE actions.** Depth 4 assigns CSV generation from the parsed file to depth 5, which passes the same task to depth 6.
26. **Step 38, depth 6 – EXEC.** Write a sorted CSV from the parsed records, with validity flags. Missing fields would be written empty rather than filled from averages. Exit code: 0.
27. **Steps 39–40, depths 6 and 5 – two RETURN actions.** Report completion of the sorted CSV and quality flags.
28. **Steps 41–42 – two CREATE actions.** Depth 4 asks depth 5 for the quality report from nine parsed records, calling the temperature field `temp`. Depth 5 passes this instruction to depth 6.
29. **Step 43, depth 6 – EXEC.** Attempt report generation with the nonexistent field `temp`. The script also assigns the full dataset size to each device's record count. Exit code: 1.
30. **Step 44, depth 6 – RETURN.** Report `KeyError: 'temp'` and request inspection of the records.
31. **Step 45, depth 5 – CREATE a new depth-6 worker.** Request the first three parsed records and their field names and types.
32. **Step 46, depth 6 – EXEC.** Read and pretty-print those records. Exit code: 0.
33. **Step 47, depth 6 – RETURN.** Identify `device_id`, `timestamp`, `temperature`, and `humidity`; note that some timestamps are invalid.
34. **Steps 48–56, depth 5 – nine identical CREATE attempts.** Repeat the report-generation instruction, still using `temp`. Each attempt is rejected with `node_id_already_used`; no child runs.
35. **Step 57, depth 5 – RETURN.** Report `KeyError: 'temperature'`, while also stating that the script used `temp` and that inspection confirmed `temperature`.
36. **Steps 58–63, depth 4 – six identical CREATE attempts.** Request report generation with the corrected field name `temperature`, still specifying nine records. Each attempt is again rejected with `node_id_already_used`. The trace ends without another worker execution or a final root answer.

Case 2B: The SFT team on the same sensor task (continued)

F / Observed outcome. The team exhausts its 64-action budget and passes 2/7 tests (reward 0). Of 21 executed commands, 15 exit with a nonzero code. Both this team and the successful team reach depth 6; their observed difference is the execution and handling of intermediate outputs, not maximum depth.

Case 3: A complete return chain carries an incorrect calculation

Q / Nemotron development task 0716. Process `raw_tickets.jsonl` into category and customer summaries and a processing log. Filter missing customer IDs, parse timestamps, represent unresolved tickets by -1, and compute the requested counts, averages, resolution rates, and urgency scores. Priority weights are 3 for critical, 2 for high, 1.5 for medium, and 1 for low.

Trajectory / LACA-DAPO team.

1. **Step 0, Root – DELEGATE to depth 1.** Request a Python analysis module and its execution, passing the task requirements to the child.
2. **Step 1, depth 1 – CREATE depth 2.** Request the analysis script and output directory.
3. **Step 2, depth 2 – CREATE depth 3.** Request creation and execution of the analysis script.
4. **Step 3, depth 3 – CREATE depth 4.** Pass on the script-and-execution task with the task requirements.
5. **Step 4, depth 4 – CREATE depth 5.** Request creation of the output directory and a check that the input exists.
6. **Step 5, depth 5 – CREATE the first depth-6 worker.** Assign output-directory creation.
7. **Step 6, the first depth-6 worker – EXEC.** Run `mkdir -p /app/output`. Exit code: 0.
8. **Step 7, the first depth-6 worker – RETURN.** Report that the output directory was created.
9. **Step 8, depth 5 – CREATE the second depth-6 worker.** Assign an input-file existence check.
10. **Step 9, the second depth-6 worker – EXEC.** Check whether `raw_tickets.jsonl` exists. Exit code: 0.
11. **Step 10, the second depth-6 worker – RETURN.** Report that the input file exists.
12. **Step 11, depth 5 – CREATE the third depth-6 worker.** Request the complete analysis script and all three output files.
13. **Step 12, the third depth-6 worker – EXEC.** Write and run the script. It filters and transforms records, groups by category and customer, and writes both JSON summaries and the log. For urgency, it looks up the *category* in the priority-weight mapping and falls back to weight 1, then multiplies by the category's ticket count. The command exits with code 0 and reports 15 input tickets, one filtered ticket, four categories, and seven customers.
14. **Step 13, the third depth-6 worker – RETURN.** Report that the script ran and the three outputs were generated.
15. **Steps 14–18, depths 5, 4, 3, 2, and 1 – five RETURN actions.** Relay completion and the reported processing counts. The depth-1 return also asserts that the schemas and calculations are correct.
16. **Step 19, Root – RETURN.** Report that all outputs were generated and verified. The root executes no command after receiving the child result.

F / Observed outcome. Four of five supplied tests pass, but the task reward is 0. The failing calculation has urgency 4 where the verifier expects 7. All three commands exit normally and the return chain completes; the numerical error remains in the final output.