Learning Before the First Success: Bidirectional Progress-Guided Training from Failed Cyber Interactions

Anonymous authors
Paper under double-blind review

Abstract

Training large language model (LLM) agents on difficult interactive tasks faces a bootstrapping barrier: when no rollout succeeds, terminal rewards make distinct failures indistinguishable. We study how failed interactions can still drive learning before the first complete solution. We introduce Bi-PGSD, a bidirectional progress-guided framework that pairs execution-verified forward effects with backward goal reasoning to construct locally checkable objectives. These objectives provide environment-anchored rewards for policy updates; target-side validation enables cross-task reuse; and, after a complete path is independently reproduced, retrieval and intermediate guidance are progressively withdrawn to consolidate autonomous behavior. Under a fixed computation budget, Qwen3.5-9B on Cybench strict zero-success (Z) improves first-success discovery from 50.0% with strong-summary retrieval to 62.5%. On scaffold-free Cybench family-held-out tasks, the trained policy improves complete-task success by 10.3 percentage points over the initial policy. Across mechanism and transfer controls, the results show that failed trajectories become useful training signals when local effects are both verified and directed toward the terminal goal.

1 Introduction

Large language model (LLM) agents offer a way to scale vulnerability discovery and security assessment through tool-based interaction (Zhang et al., 2025; The Big Sleep Team, 2024). Training them on difficult cyber tasks, however, faces a bootstrapping problem: successful-trajectory training requires solutions that the current policy may not yet discover. Existing solved examples may not apply to a new program, configuration, or task, leaving an exploration budget filled with failed attempts. How can an agent learn from these attempts before it discovers its first complete solution?

The obstacle is not simply a lack of interaction, but a lack of useful distinctions in the training signal. Under a binary terminal reward, attempts with different local outcomes receive the same failure label; an all-failure group has no reward-derived relative advantage (Yu et al., 2025). Writeup-based synthesis and execution-verified trajectory collection provide valuable cyber training data (Zhuo et al., 2026; 2025), but do not remove this bottleneck for tasks outside their coverage. Hindsight goal relabeling, reflection memories, and skill libraries provide foundations for reusing experience (Andrychowicz et al., 2017; Shinn et al., 2023; Wang et al., 2023). This leads to a sharper question: how can failed experience become both reliable enough to learn from and relevant to the original goal?

Task-level failure does not imply that every interaction is useless. Section 3 examines local effects—what changed, whether it can be reproduced, and whether reuse alters later exploration. Verified forward effects establish what the agent can do, but not which capabilities matter for the current goal. Our design principle is therefore to use forward effects as evidence and backward goal reasoning to propose what to learn next.

Bi-PGSD stores action effects with applicability conditions and execution evidence, links them to goal-regressed candidate conditions, and promotes candidates to local objectives

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.

only after environment testing. Fresh goal-conditioned trajectories provide verified rewards for policy updates; the fixed terminal verifier alone defines complete success.

Across tasks, stalled tasks retain unmet conditions and reuse external experience only after target-side validation, allowing capabilities to accumulate before complete success without supplied demonstrations or manual decompositions. Independently reproduced successes then support training while retrieval and intermediate guidance are progressively withdrawn, moving from discovery to reproduction to independent generation.

The evaluation follows the same progression across six cyber benchmark families and seven tracks: first-success discovery, cross-task reuse, and scaffold-free retention. On Qwen3.5-9B in Cybench-Z, first-success discovery rises from 50.0% with strong-summary retrieval to 62.5%; on scaffold-free Cybench family-held-out tasks, the trained policy improves complete-task success by 10.3 percentage points over the initial policy.

Our contributions are: 1) a failure-to-learning formulation separating verified local effects from terminal success; 2) Bi-PGSD, connecting verified experience, backward goal reasoning, local updates, target-validated sharing, and progressive guidance withdrawal; and 3) a staged evaluation of first-success discovery, cross-task reuse, and scaffold-free held-out capability with matched controls and mechanism interventions.

2 Related Work

Cyber agents and training. EnIGMA improves cyber-task solving through interactive tools and interfaces (Abramovich et al., 2025). Cyber-Zero synthesizes trajectories from challenge writeups, while CTF-Dojo provides executable environments and execution-verified training trajectories (Zhuo et al., 2026; 2025). Bi-PGSD complements these advances by constructing verifiable local training objectives when the training pool has produced no complete successful trajectory.

Learning from unsuccessful experience. HER reinterprets experience using goals actually achieved (Andrychowicz et al., 2017). ETO learns trajectory preferences from exploration, while NAT incorporates successful and failed trajectories into agent fine-tuning (Song et al., 2024; Wang et al., 2025). Reflexion and Voyager reuse feedback through verbal memories or executable skill libraries (Shinn et al., 2023; Wang et al., 2023). Our contribution connects verified, conditional action effects to the original task goal, turning failure experience into local policy-learning signals and sharing useful knowledge across tasks before complete success.

Goal-directed exploration and curricula. Go-Explore preserves and revisits promising states, while reverse curriculum generation expands training outward from a known goal state (Ecoffet et al., 2021; Florensa et al., 2017). Bi-PGSD constructs its curriculum from goal descriptions and verified forward effects, using backward reasoning to propose locally testable conditions and coordinating the acquisition of relevant skills across unsolved cyber tasks.

3 Before the First Success: Learning from Unsuccessful Interaction

Building on the introduction's first-success problem, we ask how unsuccessful interactions can support cumulative progress under sparse terminal rewards. Three groups of experiments examine local transition information, its reuse across attempts, and goal-conditioned proposals with execution-based validation; together, they motivate the subsequent method.

We use a strict empirical zero-success setting: tasks with no terminal success in 16 independent screening attempts. This finite-screening condition does not assert zero underlying success probability. Agents retain the goal, observable feedback, a terminal verifier, and base-model priors, but receive no successful demonstrations, human subgoals, or process rewards. Conditions share interfaces and budget caps, including replay and validation costs. Except for the terminal RL baseline, weights remain fixed. We measure first success within

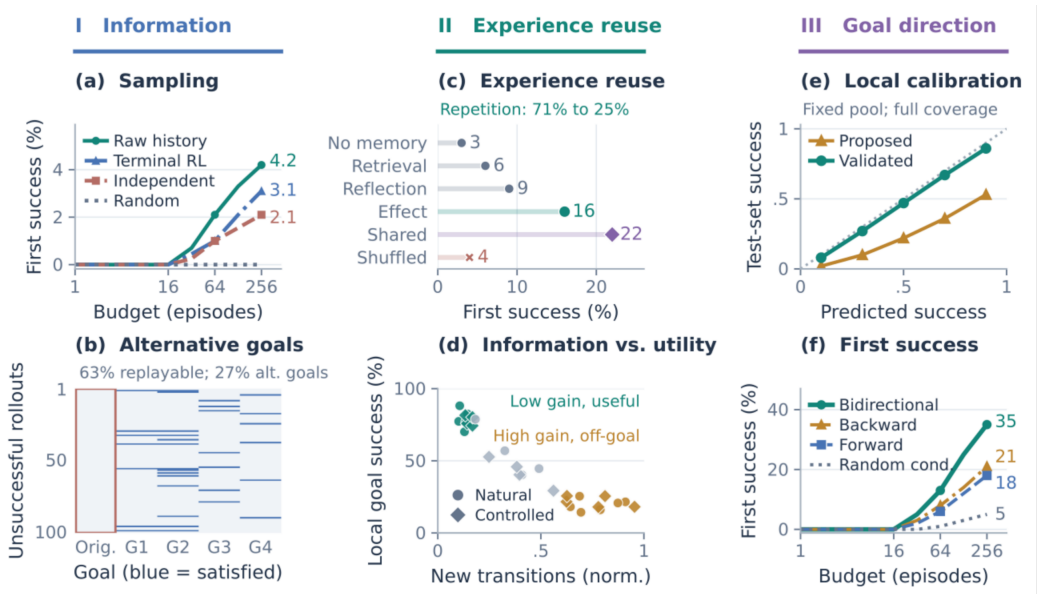


Figure 1: Before the first success: three controlled comparisons, read by column. (a,b) Sampling and alternative-goal reevaluation. (c,d) Experience reuse and knowledge growth versus independently evaluated local utility; Effect denotes validated transition memory. (e,f) Local-success calibration on a fixed candidate pool and exploration direction. First success requires a complete, terminally verified solution; local success does not.

budget and reproducible local effects; Appendix A.1 specifies screening, resource accounting, and statistics.

3.1 Failed trajectories contain reusable transition information

To test whether more rollouts overcome stagnation, we compare legal-action random sampling, independent model rollouts, terminal-reward RL, and raw-history retention, with up to 256 episodes per task. Tracking repetition alongside completion distinguishes additional search effort from reuse of familiar transitions. Figure 1a reports endpoint first-success rates of 2.1%, 3.1%, and 4.2% for the latter three, alongside an increase in repeated transitions. For a GRPO baseline (Shao et al., 2024b), all-failure groups have zero relative advantage for the binary terminal-reward term: local discoveries receive no distinction through that term. Appendix A.2 specifies reward-group, repetition, and search-branch diagnostics.

Low terminal success need not imply uninformative interaction. From the same unsuccessful trajectories, we extract records (c, a, o): action a produces outcome o under context c. Context may include observation history, not a complete Markov state. We check state differences, replay the action sequence, and reevaluate outcomes against independently defined tasks or predefined goal predicates, following the goal-conditioned reevaluation principle of Andrychowicz et al. (2017). Figure 1b reports 63% of unsuccessful trajectories containing reproducible effects and 27% satisfying an alternative goal. These trajectory-level proportions can overlap and do not establish progress toward the original goal.

Terminal success, $R_g(\tau) = \mathbf{1}\{\text{Verify}(g, \tau) = 1\}$, evaluates a trajectory against goal g; local records describe what happened. Reproducible effects could therefore survive a change of goal, provided their applicability is rechecked when conditions or environments change. We next test whether retaining them changes subsequent decisions.

3.2 Reusing experience improves adaptation, but information gain need not imply goal progress

To test cumulative reuse, we compare no memory, episodic retrieval, verbal reflection, and validated transition memory (Effect). Reflection and cross-task experience reuse motivate these baselines (Shinn et al., 2023; Zhao et al., 2024), without assuming identical imple-

mentations. Transition memory stores conditions, actions, outcomes, and supporting or conflicting evidence, checking applicability during retrieval. Equal-length irrelevant records and shuffled condition–outcome bindings control for extra context; all conditions share a context budget. This comparison isolates whether organizing experience changes how the same model acts on later attempts.

The pattern in Figure 1c shows useful reuse: repetition falls from 71% to 25%, while first success rises from 3% without memory to 16% with Effect, 22% with cross-training-task sharing, but only 4% after shuffling. Necessary confirmation replays are accounted for separately from avoidable repetitions. Appendix A.3 specifies pre-success knowledge growth, subsequent behavior, cross-task reuse, negative transfer, and irrelevant-context controls. Such differences implicate applicable transition information rather than context length alone. With fixed weights, they show system adaptation, not parameter learning.

Does accumulating knowledge also identify where to explore? Holding memory fixed, we select starting states randomly or by novelty, newly validated transitions, or short-horizon reachability. An isolated evaluator measures downstream utility. Figure 1d shows high knowledge gain with low local utility and familiar states with high utility; its gain score counts new records, not information-theoretic information gain. Appendix A.4 specifies ranking comparisons and controlled states. These patterns show that expanding exploration need not prioritize conditions useful for the current goal.

The remaining question is therefore directional: evidence about what actions do does not by itself identify what the current goal may require. We examine whether reasoning backward from the goal supplies candidate enabling conditions without human task decomposition.

3.3 Goal-directed subgoal proposals benefit from execution-based validation

To test how goal information shapes backward proposals, we compare subgoal proposals using only terminal judgments, adding goal semantics, and additionally using existing transition knowledge. The terminal-only condition is a diagnostic ablation that withholds goal semantics. Appendix A.5 tests whether semantics improve relevance and experience reduces unreachable proposals, while errors still accumulate with backward depth. Proposed subgoals are checkable candidate enabling conditions, not established necessities. Model-based planning and local subgoal models (Tang et al., 2024; Lo et al., 2022) motivate checking these proposals against the environment rather than trusting semantic plausibility.

We first test whether each condition can be reached from an actual state and repeatedly achieved, then evaluate its utility for a testable local successor goal. Figure 1e reports the relationship that this evidence improves local-success calibration on an independent test set, using a fixed candidate pool and reporting prediction coverage. Calibration asks whether predicted local completion probabilities agree with held-out execution frequencies. Before first success, such tests support local applicability, not an entire solution. Conditions without testable successors remain unresolved; the terminal verifier alone determines complete success.

With the same transition memory and budgets, we compare forward exploration, direct backward-condition exploration, random conditions, and bidirectional exploration filtered by forward execution evidence. Figure 1f reports first-success rates of 18%, 21%, 5%, and 35%, respectively; the denominator includes runs unsuccessful at budget exhaustion. Appendix A.6 specifies condition quality, reachability, local interventions, and forward–backward state compatibility to distinguish relevance, executability, and downstream utility.

Together, the patterns motivate retaining conditional effects, reusing them across attempts, and combining goal-directed proposals with execution evidence. Based on these observations, we develop bidirectional progress-guided exploration in Section 4, maintaining revisable transition knowledge and candidate conditions. Local evidence selects the next exploration step; executable connections and terminal verification test complete solutions. Subsequent experiments evaluate the full method, parameter learning after success, and retention after memory removal.

4 Method

The observations in Section 3 imply two requirements for pre-success learning: preserve conditional effects with execution evidence, and let the task goal select which effects matter. Bi-PGSD closes this loop by storing verified experience, regressing unresolved goals to locally testable conditions, and consolidating independently reproduced successes into an autonomous policy. Shared memory and parameters connect discovery to consolidation; Figure 2 summarizes the resulting progression.

4.1 Cyber Task Formulation

We study executable CTF and controlled-vulnerability tasks $\mathcal{T} = \{T_i\}_{i=1}^{N}$ in public, isolated, authorized environments. Task T_i specifies an initial distribution μ_i, permitted materials, goal g_i, tools $\mathcal{A}_i$, and a fixed terminal grader V_i, without assuming a common solution structure across cyber domains (Zhang et al., 2025; Wang et al., 2026).

At each decision, the shared policy conditions on the visible action–observation history and optional guidance:

$$h_t = (o_0, a_0, o_1, \ldots, a_{t-1}, o_t), \qquad a_t \sim \pi_\theta(\cdot \mid h_t, g_i, c_t). \tag{1}$$

Here o_0 contains permitted materials, o_t execution feedback, and c_t retrieved experience or local prompts. Evidence is bound to version, configuration, session, permissions, and objects. Only V_i defines complete success; exits, HTTP responses, and exceptions are local evidence. A zero-success start means no success within screening budget K_0, not zero probability. We log accessible positives, original-state success, independent reproduction, and library admission; Appendices C.1 and C.4 fix provenance.

4.2 Learning Reusable Experience from Cyber Interactions

Following Section 3.1, Bi-PGSD keeps a scoped store $\mathcal{M}$ of records

$$k = (\xi, P, \omega, E, V_E, \mathcal{L}). \tag{2}$$

The scope ξ identifies software, interface, location, and objects; P gives observable preconditions; ω is an operation or short skill; E its effect; V_E its checker; and $\mathcal{L}$ provenance and conflicts. The record therefore supports option-style reuse without dropping execution evidence (Andrychowicz et al., 2017; Sutton et al., 1999; Konidaris et al., 2018).

Parsing a response or reaching a service does not prove a vulnerability. Each proposal is bound to a native checker, re-executed in the current environment, and admitted only after independent fixed-size repetition; conflicts remain stored. Complete successes enter $\mathcal{D}^+$ with ordered traces, whereas failures contribute verified effects, skills, and conditional counterexamples. Policy θ, predictor q_ϕ, records, and checkers update separately while V_i stays fixed.

4.3 Goal-Directed Bidirectional Exploration and Task Scheduling

To operationalize Section 3.3, Bi-PGSD combines backward hypotheses with forward execution. For an unresolved goal, permitted materials, visible history, and shared experience produce bounded candidate condition sets: sets are alternative routes (OR), while conditions within a set must hold jointly (AND). Candidates are checked and executed in the current version, object, session, and location; state changes trigger rechecking, and a fixed-budget parent continuation separates usefulness from truth. These relations remain revisable hypotheses, not command sequences (Fikes & Nilsson, 1971); Appendix C.4 gives the interventions.

For an outcome p that has been checked and reproduced in the current task, we construct a local goal-conditioned task and binary achievement reward:

$$b = (\mu_b, p, H_b, \mathcal{A}_i, V_p), \qquad R_b(\tau) = \mathbf{1}[\exists\, t \le H_b :\ V_p(h_t) = 1]\,. \tag{3}$$

The start distribution μ_b covers reachable contexts with $V_p(h_0) = 0$ and horizon H_b; trials stop on achievement or exhaustion. State predicates and evidence goals follow Appendix C.1, missing feedback remains unknown, and clean reset plus real-prefix replay refreshes bindings. Checkpoints, restoration, and checking are charged under return-then-explore (Ecoffet et al., 2021).

We update the policy with KL-regularized REINFORCE leave-one-out (RLOO) (Ahmadian et al., 2024). For $G \geq 2$ fresh, independent trajectories under the same local goal, start-sampling rule, and fixed batch configuration, the reward advantage and its policy-gradient estimate are

$$\widehat{A}_k^R = R_b(\tau_k) - \frac{1}{G-1}\sum_{j\neq k} R_b(\tau_j),$$
$$\widehat{g}_b^R = \frac{1}{G}\sum_{k=1}^{G} \widehat{A}_k^R \sum_t \nabla_\theta \log \pi_\theta(a_t^k \mid h_t^k, p, c_t^k). \tag{4}$$

Equal terminal rewards give zero reward advantages, while reachable local tasks can separate behavior; this does not guarantee higher terminal success (Appendix B.1). KL anchors a fixed reference. Appendix C.2 specifies the objective and unknown-label handling; replay supplies starts and fresh policy trajectories supply on-policy updates.

Across tasks, a progress curriculum allocates effort by local improvement and execution cost (Graves et al., 2017). Stalled tasks save unmet conditions; relevant experience triggers target-side verification returns, periodic retries preserve opportunity, and a reserved fraction maintains original-state attempts. Appendix C.3 defines the schedule, detour costs, and observability limits.

4.4 Consolidating Successful Trajectories into Autonomous Policies

Because a complete path does not imply autonomous generation, an initial-state success enters $\mathcal{D}^+$ only after independent clean-instance reproduction. Self-generated prefixes are re-executed before new behavior; script replay, neural autonomy, and cached answers remain distinct. Appendices C.3 and C.4 define this gate.

We combine supervised fine-tuning on complete successes with terminal-reward RL on fresh trajectories. Success filtering follows the self-training principle of ReST$^{\mathrm{EM}}$ (Singh et al., 2024), with each demonstration retaining the inputs actually visible when its actions were chosen:

$$\mathcal{L}_{\mathrm{SFT}}(\theta) = -\mathbb{E}_{(i,\tau,c)\sim\mathcal{D}^+}\left[\sum_t m_t \log \pi_\theta(a_t \mid h_t, g_i, c_t)\right]. \tag{5}$$

The mask m_t keeps supported, interface-expressible actions and excludes environment outputs, controller-only operations, and direct answers; historical demonstrations are not relabeled as fresh on-policy data (Appendix C.3).

The curriculum moves from short suffixes to the original initial state, then withdraws prompts and retrieval; fresh trajectories and a prespecified criterion limit distribution shift (Florensa et al., 2017; Ross et al., 2011). Appendices C.3 and C.4 compare immediate versus progressive withdrawal, retention, and held-out transfer.

Method-to-evaluation bridge. The resulting mechanism-to-outcome chain is explicit: verified effects should reduce avoidable repetition and enable transfer, backward goal direction should make shared experience useful for the original task, and progressive withdrawal should reveal whether the capability persists without external support. The experiments therefore report first-success discovery, cost and transfer, and scaffold-free held-out performance as linked tests of one framework rather than isolated improvements.

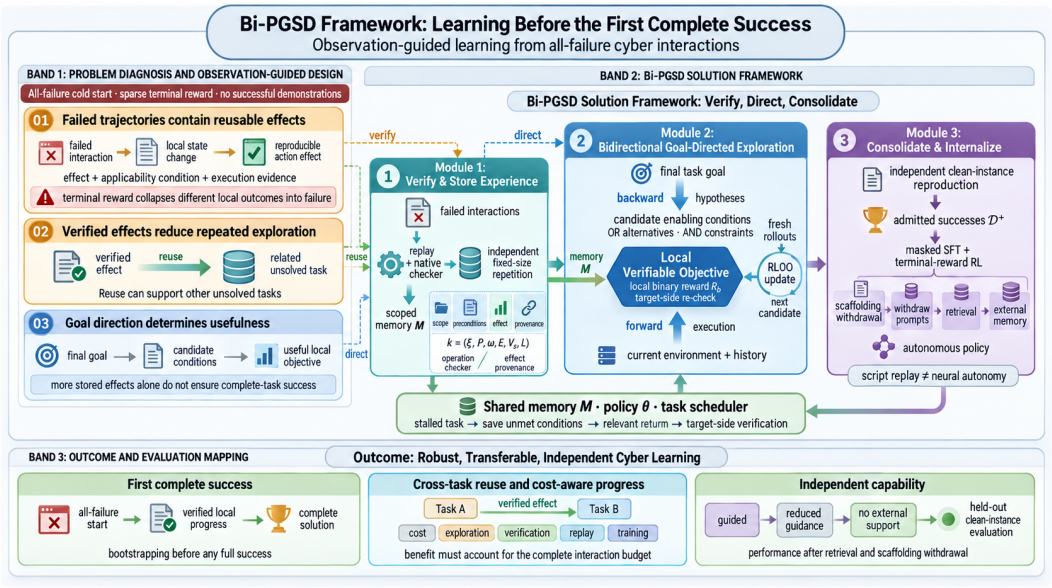


Figure 2: Framework. Top: the strict zero-success bottleneck and three observations—failures expose reusable effects, verification reduces repetition, and goal direction selects useful effects. Middle: verified memory, backward hypotheses, and forward execution form local objectives; independently reproduced successes are consolidated as scaffolds are withdrawn. Arrows/icons show evidence and feedback without adding data. Bottom: first success, budget-aware transfer, and held-out capability.

5 Experiments

Experiments ask whether failure-driven learning can (i) reach first complete success under matched end-to-end cost, (ii) transfer verified experience across tasks, and (iii) consolidate success into scaffold-free capability.

Protocol summary. Across six benchmarks—Cybench, NYU CTF Bench, InterCode-CTF, CVE-Bench, CyberGym, and SEC-bench Pro (Zhang et al., 2025; Shao et al., 2024a; Yang et al., 2023; Zhu et al., 2025; Wang et al., 2026; Lee et al., 2026)—we evaluate strict zero-success (Z) and success-assisted (A) conditions under common task boundaries and end-to-end budgets; frozen held-out trials restart from the original initial state and forbid cross-episode learning. Appendices D.1, D.2, and I specify the splits and update boundaries, cost/statistical accounting, and information-boundary audit conditions.

5.1 Performance and Cost

Failure experience can produce first success. Table 1 tests whether failed experience can compose a complete path while the fixed terminal verifier alone defines success. Qwen3.5-9B on Cybench-Z rises from 50.0% with strong-summary retrieval to 62.5%; InterCode is near saturation and some cells are flat or negative. Table 4 reports the screened task counts and common zero-success intersections.

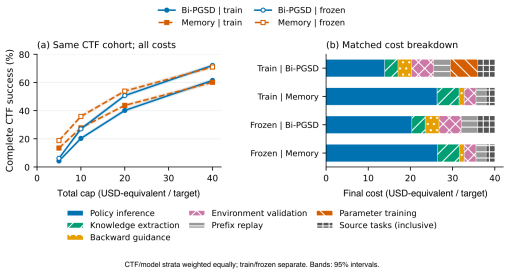
The gain must repay its cost. With matched end-to-end budgets (Figure 3a), the CTF mean across six trainable-policy candidates is only 1.4 points above strong-summary retrieval; maintenance consumes 55.2% of cost, and Bi-PGSD trails at low budgets. Figure 5 gives per-model A-condition curves; Appendix D.2 gives cost and statistical accounting.

5.2 Experience and Cross-Task Transfer

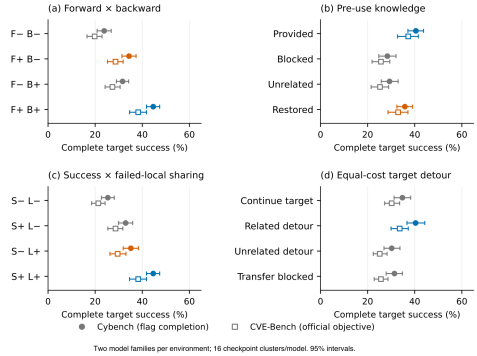
Transfer needs both experience and goal conditions. Figure 3b (top) intervenes on forward experience and applicability conditions. On Cybench, their interaction is 2.5 points with an interval crossing zero, while release-before-use exceeds blocking by 12.1 points; the evidence supports complementary rather than super-additive effects.

Table 1: Complete-task success (%), Bi-PGSD / strong-summary retrieval. Z: 24 pools × 4 targets; A: 24 × 16, with distinct denominators. 40.0 dollars per target-equivalent unit; upper block updates policy, lower block freezes it, and SEC projects remain separate.

Model	Info.	Cybench	NYU	InterCode	CVE	CyberGym	SEC V8	SEC SM
Trainable-policy candidates; Bi denotes Bi-PGSD-Full								
Qwen3.5-9B	Z	62.5/50.0	70.8/66.7	100.0/100.0	66.7/54.2	70.8/66.7	62.5/33.3	54.2/41.7
	A	46.4/43.0	44.5/39.3	83.9/91.7	50.0/44.8	42.7/37.0	37.8/31.2	39.3/32.3
Qwen3.5-27B	Z	79.2/66.7	70.8/58.3	100.0/100.0	83.3/70.8	54.2/50.0	83.3/75.0	54.2/41.7
	A	54.2/46.6	51.3/42.4	90.4/96.6	58.6/49.5	48.4/43.2	49.0/40.1	44.8/39.1
Qwen3.8-27B	Z	87.5/79.2	66.7/45.8	100.0/100.0	91.7/66.7	70.8/50.0	50.0/33.3	62.5/58.3
	A	58.9/52.1	60.9/53.4	87.0/94.3	56.8/51.8	47.1/39.3	50.3/42.4	50.0/41.4
GLM-4.7-Flash	Z	70.8/50.0	83.3/70.8	87.5/91.7	70.8/58.3	54.2/37.5	70.8/62.5	50.0/25.0
	A	48.2/42.4	46.9/40.6	78.6/89.3	55.5/48.4	40.4/34.4	41.1/32.0	35.9/30.7
Gemma 4 26B-A4B	Z	75.0/66.7	58.3/37.5	95.8/95.8	75.0/62.5	75.0/66.7	79.2/58.3	66.7/62.5
	A	52.9/46.4	50.0/39.8	79.7/93.5	50.0/40.1	45.8/40.4	44.5/37.5	47.4/40.1
Frozen-policy frontier candidates; Bi denotes Bi-PGSD-Frozen								
GPT-6 Astra	Z	79.2/70.8	91.7/83.3	95.8/100.0	91.7/87.5	83.3/79.2	87.5/75.0	83.3/70.8
	A	65.4/60.4	68.0/62.0	92.4/97.4	72.1/65.6	57.8/52.6	60.9/52.9	58.1/53.6
Claude Opus 5.5	Z	83.3/75.0	87.5/75.0	100.0/100.0	83.3/70.8	87.5/87.5	87.5/87.5	87.5/75.0
	A	73.7/68.5	69.5/66.9	92.4/97.7	66.9/64.3	63.8/58.9	63.5/58.9	62.5/57.6
Gemini 3.8 Flash	Z	87.5/79.2	75.0/66.7	95.8/95.8	75.0/50.0	50.0/41.7	70.8/62.5	66.7/37.5
	A	62.5/55.5	51.8/47.4	90.9/98.2	58.6/52.3	49.7/42.7	49.7/43.5	53.9/48.2
DeepSeek-V4.1-Flash	Z	66.7/54.2	91.7/87.5	100.0/100.0	79.2/70.8	83.3/75.0	62.5/62.5	87.5/79.2
	A	58.6/54.4	58.6/55.2	91.7/99.0	57.6/52.9	48.2/44.8	46.9/41.9	50.5/43.8
GLM-5.3-Flash	Z	79.2/70.8	75.0/62.5	100.0/100.0	75.0/66.7	62.5/37.5	83.3/75.0	66.7/62.5
	A	65.9/62.5	58.3/52.1	85.4/94.0	58.3/53.6	55.5/50.5	51.3/46.4	55.5/51.3
Kimi K3	Z	70.8/70.8	79.2/70.8	100.0/100.0	91.7/83.3	66.7/54.2	66.7/62.5	79.2/66.7
	A	62.5/56.5	62.0/54.2	87.5/94.8	63.0/57.0	61.7/56.0	49.7/46.1	51.8/45.8



(a) Effects and end-to-end cost. Under A, complete-task success in three CTF environments and seven mutually exclusive terminal-budget cost components; source, extraction, guidance, verification, replay, and training are charged. Bands: pointwise 95% stratified-bootstrap intervals over 24 pools per model/environment.



(b) Experience and transfer. Top: forward experience F, backward conditions B, and knowledge release/blocking/replacement/recovery. Bottom: complete success S, failed local experience L, and equal-cost detours. Two model families/environment, 16 source checkpoints/level; endpoint is complete-target success with 95% stratified-pool intervals.

Figure 3: Main cost and transfer diagnostics. The paired panels summarize the budget and mechanism tests that connect the observations to the experimental claims.

Shared experience must alter later attempts. Figure 3b (bottom) compares complete success, validated failed fragments, and equal-cost detours. On Cybench, the sharing interaction is 2.0 points with an interval crossing zero, while a relevant detour exceeds direct continuation by 5.6 points. Irrelevant-source and transfer-blocking controls rule out extra trials alone; because source-task success is allowed, these arms test transfer rather than Z discovery. Appendix F gives transfer controls.

Table 2: Component ablation. Matched mechanism pools and budget; two model families per environment. Values are complete-task success with 95% intervals.

Configuration	Cybench (%)	CVE-Bench (%)
Bi-PGSD	44.7 [41.8, 47.4]	38.3 [34.6, 41.8]
Strong-summary retrieval	38.2 [34.8, 41.5]	32.6 [29.2, 35.8]
Frozen local checker	38.0 [35.2, 40.8]	31.9 [28.6, 35.4]
Frozen guidance	38.3 [35.3, 41.1]	32.4 [29.4, 35.7]
Filtered local SFT	41.4 [38.3, 44.8]	35.5 [32.3, 38.9]
No local policy updates	34.1 [31.2, 37.1]	28.9 [25.8, 32.3]
Whole failed traces shared	36.1 [33.1, 38.8]	30.3 [27.0, 33.5]
Round-robin scheduling	43.4 [40.4, 46.1]	37.8 [34.0, 41.4]

Brackets report pooled/checkpoint bootstrap intervals.

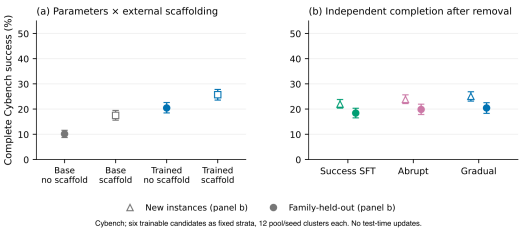


Figure 4: From support to independent completion. (a) Base/post-training parameters with/without scaffolding on Cybench family-held-out tasks; (b) scaffold-free pass@1 under successful SFT, immediate withdrawal, or progressive withdrawal. Six candidates; 95% stratified-pool intervals.

Ablations isolate components. Table 2 separates memory, checker/guidance, local learning, and scheduling. On Cybench, Bi-PGSD is 1.3 points above round-robin and the updated checker has a 2.3% false-positive rate, showing that scheduling is not the primary source and local feedback does not imply a complete path. Further controls are in Appendices E and H.

5.3 From Success to Independent Capability

Discovery, replay, and internalization are distinct. Independent resets separate script replay from scaffold-free regeneration: in E24 (Appendix G), Cybench replay reaches 77.6% but regeneration only 25.9%. Figure 4(a) then shows a 10.3-point gain after scaffold-free parameter training under fixed evaluation parameters, sealed learned knowledge, and original-state restarts.

Independent capability must transfer to new tasks. Figure 4(b) evaluates withdrawal on same-family new and family-held-out tasks. On the Cybench family holdout, progressive withdrawal differs from successful SFT by 2.0 points. Appendix G reports retention, trajectory matching, other environments, and cross-cyber adaptation; cost and withdrawal use independent cohorts.

6 Conclusion

Bi-PGSD turns unsuccessful cyber-agent interactions into learning signals by combining verified forward effects with backward goal reasoning. The resulting local objectives support pre-success learning and target-validated cross-task reuse, while independently reproduced successes are consolidated through progressive guidance withdrawal. The experiments connect these mechanisms to first-success discovery, transfer, and scaffold-free capability.

AI Use Statement

Generative AI tools were used for coding assistance, drafting and revising portions of the manuscript, and language editing and polishing. All AI-assisted code and text were reviewed by the authors, and the authors verified the reported experiments, numerical results, citations, and final manuscript. The authors take full responsibility for the content of this work.

References

Talor Abramovich, Meet Udeshi, Minghao Shao, Kilian Lieret, Haoran Xi, Kimberly Milner, Sofija Jancheska, John Yang, Carlos E. Jimenez, Farshad Khorrami, Prashanth Krishnamurthy, Brendan Dolan-Gavitt, Muhammad Shafique, Karthik R. Narasimhan, Ramesh Karri, and Ofir Press. EnIGMA: Interactive tools substantially assist LM agents in finding

security vulnerabilities. In Proceedings of the 42nd International Conference on Machine Learning, volume 267 of Proceedings of Machine Learning Research, pp. 246–355. PMLR, 2025. URL https://proceedings.mlr.press/v267/abramovich25a.html. Published version; arXiv:2409.16165 is the same work.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 12248–12267. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.662. URL https://aclanthology.org/2024.acl-long.662/.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In Advances in Neural Information Processing Systems, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/453fadbd8a1a3af50a9df4df899537b5-Abstract.html.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. First return, then explore. Nature, 590:580–586, 2021. doi: 10.1038/s41586-020-03157-9. URL https://www.nature.com/articles/s41586-020-03157-9. Published Nature version; arXiv:2004.12919 is the same work.

Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. Artificial Intelligence, 2(3–4):189–208, 1971. URL https://www.sciencedirect.com/science/article/pii/0004370271900105.

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In Proceedings of the 1st Annual Conference on Robot Learning, volume 78 of Proceedings of Machine Learning Research, pp. 482–495. PMLR, 2017. URL https://proceedings.mlr.press/v78/florensa17a.html.

Alex Graves, Marc G. Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In Proceedings of the 34th International Conference on Machine Learning, volume 70 of Proceedings of Machine Learning Research, pp. 1311–1320. PMLR, 2017. URL https://proceedings.mlr.press/v70/graves17a.html.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association, 58(301):13–30, 1963. doi: 10.1080/01621459.1963.10500830. URL https://www.jstor.org/stable/2282952.

George Konidaris, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. From skills to symbols: Learning symbolic representations for abstract high-level planning. Journal of Artificial Intelligence Research, 61:215–289, 2018. doi: 10.1613/jair.5575. URL https://jair.org/index.php/jair/article/view/11175.

Hwiwon Lee, Jiawei Liu, Dongjun Kim, Ziqi Zhang, Chunqiu Steven Xia, and Lingming Zhang. SEC-bench Pro: Can language models solve long-horizon software security tasks? arXiv:2605.26548v1, 2026. URL https://arxiv.org/abs/2605.26548v1. Numerical source: version 1, Table 3; not the expanded version 2 collection. Accessed 2026-09-25.

Chunlok Lo, Kevin Roice, Parham Mohammad Panahi, Scott Jordan, Adam White, Gabor Mihucz, Farzane Aminmansour, and Martha White. Goal-space planning with subgoal models. arXiv preprint arXiv:2206.02902, 2022. URL https://arxiv.org/abs/2206.02902. Version 5 revised in 2024.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, volume 15 of Proceedings of Machine Learning Research, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

Minghao Shao, Sofija Jancheska, Meet Udeshi, Brendan Dolan-Gavitt, Haoran Xi, Kimberly Milner, Boyuan Chen, Max Yin, Siddharth Garg, Prashanth Krishnamurthy, Farshad Khorrami, Ramesh Karri, and Muhammad Shafique. NYU CTF Bench: A scalable open-source benchmark dataset for evaluating LLMs in offensive security. In Advances in Neural Information Processing Systems, volume 37, 2024a. doi: 10.52202/079017-1832. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/69d97a6493fbf016fff0a751f253ad18-Abstract-Datasets_and_Benchmarks_Track.html.

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, 2024b. URL https://arxiv.org/abs/2402.03300.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. arXiv preprint arXiv:2303.11366, 2023. URL https://arxiv.org/abs/2303.11366.

Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J. Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron Parisi, Abhishek Kumar, Alex Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Elsayed, Hanie Sedghi, Igor Mordatch, Isabelle Simpson, Izzeddin Gur, Jasper Snoek, Jeffrey Pennington, Jiri Hron, Kathleen Kenealy, Kevin Swersky, Kshiteej Mahajan, Laura Culp, Lechao Xiao, Maxwell L. Bileschi, Noah Constant, Roman Novak, Rosanne Liu, Tris Warkentin, Yundi Qian, Yamini Bansal, Ethan Dyer, Behnam Neyshabur, Jascha Sohl-Dickstein, and Noah Fiedel. Beyond human data: Scaling self-training for problem-solving with language models. Transactions on Machine Learning Research, 2024. URL https://arxiv.org/abs/2312.06585.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization of LLM agents. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 7584–7600. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.409. URL https://aclanthology.org/2024.acl-long.409/.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. Artificial Intelligence, 112 (1–2):181–211, 1999. URL https://www.sciencedirect.com/science/article/pii/S0004370299000521.

Hao Tang, Darren Key, and Kevin Ellis. WorldCoder, a model-based LLM agent: Building world models by writing code and interacting with the environment. In Advances in Neural Information Processing Systems, 2024. URL https://arxiv.org/abs/2402.12275.

The Big Sleep Team. From naptime to Big Sleep: Using large language models to catch vulnerabilities in real-world code. Google Project Zero, November 2024. URL https://projectzero.google/2024/10/from-naptime-to-big-sleep.html.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023. URL https://arxiv.org/abs/2305.16291.

Renxi Wang, Xudong Han, Yixuan Zhang, Timothy Baldwin, and Haonan Li. NAT: Enhancing agent tuning with negative samples. In Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pp. 7385–7398. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.naacl-long.378. URL https://aclanthology.org/2025.naacl-long.378/.

Zhun Wang, Tianneng Shi, Jingxuan He, Matthew Cai, Jialin Zhang, and Dawn Song. CyberGym: Evaluating AI agents' real-world cybersecurity capabilities at scale. In International Conference on Learning Representations, 2026. URL https://iclr.cc/virtual/2026/poster/10011727. Published as an ICLR 2026 conference paper; numerical source for reproduced historical rows: arXiv:2506.02548v1, Figure 6 and Appendix C.

John Yang, Akshara Prabhakar, Karthik R. Narasimhan, and Shunyu Yao. InterCode: Standardizing and benchmarking interactive coding with execution feedback. In Advances in Neural Information Processing Systems, volume 36, 2023. doi: 10.52202/075280-1035. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/4b175d846fb008d540d233c188379ff9-Abstract.html.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan 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, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Andy K. Zhang, Neil Perry, Riya Dulepet, Joey Ji, Celeste Menders, Justin Lin, Eliot Jones, Gashon Hussein, Samantha Liu, Donovan Jasper, Pura Peetathawatchai, Ari Glenn, Vikram Sivashankar, Daniel Zamoshchin, Leo Glikbarg, Derek Askaryar, Haoxiang Yang, Aolin Zhang, Rishi Alluri, Nathan Tran, Rinnara Sangpisit, Kenny Oseleononmen, Dan Boneh, Daniel Ho, and Percy Liang. Cybench: A framework for evaluating cybersecurity capabilities and risks of language models. In International Conference on Learning Representations, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/3e9412a9c1d93810ef3ef7825115016b-Abstract-Conference.html.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, pp. 19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936. URL https://ojs.aaai.org/index.php/AAAI/article/view/29936.

Yuxuan Zhu, Antony Kellermann, Dylan Bowman, Philip Li, Akul Gupta, Adarsh Danda, Richard Fang, Conner Jensen, Eric Ihli, Jason Benn, Jet Geronimo, Avi Dhir, Sudhit Rao, Kaicheng Yu, Twm Stone, and Daniel Kang. CVE-Bench: A benchmark for AI agents' ability to exploit real-world web application vulnerabilities. In Proceedings of the 42nd International Conference on Machine Learning, volume 267 of Proceedings of Machine Learning Research, pp. 79850–79867. PMLR, 2025. URL https://proceedings.mlr.press/v267/zhu25i.html.

Terry Yue Zhuo, Dingmin Wang, Hantian Ding, Varun Kumar, and Zijian Wang. Training language model agents to find vulnerabilities with CTF-Dojo. arXiv preprint arXiv:2508.18370, 2025. URL https://arxiv.org/abs/2508.18370v2. Version 2.

Terry Yue Zhuo, Dingmin Wang, Hantian Ding, Varun Kumar, and Zijian Wang. CyberZero: Training cybersecurity agents without runtime. In International Conference on Learning Representations, 2026. URL https://openreview.net/forum?id=1gRTeAik4G. Published as an ICLR 2026 conference paper; numerical source for reproduced historical rows: arXiv:2508.00910v2, Table 3.

A Additional Details for the Observational Study

A.1 Common protocol and evaluation

The observational study uses 96 resettable tasks. A fixed initial policy screens each task with 16 independent attempts; only tasks with no observed terminal success enter the empirical zero-success pool. Evaluation uses fresh seeds, charges replay and validation to the same resource budget, and defines first success only by a complete solution verified from the native initial state. The common cap is 256 episodes per task with at most 64 tool actions per episode; uncertainty is task-clustered. These constants are sufficient to interpret the Figure 1 comparisons; implementation-specific logging fields are recorded by the run-manifest schema included with the submitted core code.

A.2 Sampling and goal-conditioned outcome evaluation

With matched task sets and budgets, independent model rollouts, terminal-reward RL, and raw-history retention reach endpoint first-success rates of 2.1%, 3.1%, and 4.2%, while repeated transitions increase. On the same unsuccessful-trajectory sample, 63% contain at least one reproducible local effect and 27% satisfy a prespecified alternative goal; these sets may overlap and neither statistic implies progress toward the original goal. For an all-zero terminal-reward group, the relative terminal-reward advantage is algebraically zero; other regularizers or losses may still update parameters.

A.3 Memory, accumulation, and cross-task reuse

Using the same underlying interaction history, validated transition memory reduces repetition from 71% to 25%. First-success rates are 3% with no memory, 16% with validated transition memory (Effect), 22% with cross-training-task sharing, and 4% after shuffling condition–outcome bindings. Necessary confirmation replays are accounted for separately from avoidable repetitions. These fixed-weight comparisons measure system adaptation through external experience, not parameter learning.

A.4 Exploration coverage and downstream utility

Holding memory and rollout budget fixed, the selection rules in Figure 1d separate coverage from downstream utility. High knowledge gain can coincide with low local utility, while familiar states can have high utility. Thus, accumulating new transition records is not itself a goal-directed exploration criterion; the isolated evaluator used for this comparison never feeds its scores or traces back to the learner.

A.5 Sources and quality of proposed subgoals

Backward proposals are compared with terminal judgments alone, with goal semantics, and with goal semantics plus eligible transition knowledge. Goal information improves relevance, prior verified experience reduces unreachable proposals, and deeper backward chains accumulate more errors. Independent execution tests then improve calibration of predicted local-success probabilities. Proposed conditions remain candidates until they are reached and tested; complete success is still determined only by the terminal verifier.

A.6 Execution-based validation and combined exploration

With identical transition memory and matched budgets, forward exploration, direct backward-condition exploration, random conditions, and bidirectional exploration achieve first-success rates of 18%, 21%, 5%, and 35%, respectively. The bidirectional condition filters goal-derived proposals through forward execution evidence and checks compatibility on actual states before terminal verification. These results motivate the method in Section 4: verified effects establish what is executable, while backward reasoning directs which capability should be pursued next.

B Conditional Analysis of the Method

This appendix explains three limited roles of local learning: obtaining discriminative rewards, reducing uncertainty through compatible evidence, and supporting continuous execution through learned skills. The arguments concern observable execution events. Assumptions about latent environment dynamics are analytical assumptions, not information available to the agent. None of the arguments establishes that a proposed condition is necessary for a cyber task, that a textual match implies reachability, or that bidirectional exploration has a universal complexity advantage.

B.1 When Local Rewards Distinguish Behavior

Fix a local task b, its start distribution, horizon, checker, and guidance rule. During one sampling batch, hold the policy version and all external components fixed. Let $\tau_1, \ldots, \tau_G$ be independent trajectories from that configuration, with $G \geq 2$ and fully observed binary rewards $R_k = R_b(\tau_k)$. Write the score of the policy-generated portion as

$$s_k = \sum_t \nabla_\theta \log \pi_\theta(a_t^k \mid h_t^k, p, c_t^k). \tag{6}$$

The sum excludes controller actions and any replayed prefix. Environment dynamics and the start sampler are independent of the differentiated parameters. A guidance generator that shares parameters with the actor is held at its sampling-time copy, so its outputs are conditioning inputs for this update. Assume differentiability, fixed policy support, and integrability sufficient to interchange expectation and differentiation.

Reward-gradient identity. Under these assumptions, the estimator in Equation 4 is unbiased for the reward gradient. For distinct trajectories $j \neq k$, independence and the score identity give

$$\begin{aligned} \mathbb{E}[R_j s_k] &= \mathbb{E}[R_j]\mathbb{E}[s_k] = 0, \\ \mathbb{E}[\widehat{g}_b^R] &= \mathbb{E}[R_b(\tau)s(\tau)] = \nabla_\theta \mathbb{E}[R_b(\tau)]. \end{aligned} \tag{7}$$

This is the standard leave-one-out baseline argument applied to interaction trajectories (Ahmadian et al., 2024). If all rewards in a realized group equal the same value, every reward advantage is zero. This includes both all-failure and all-success groups.

For independent trials with local success probability $p_b = \Pr(R_b(\tau) = 1)$, the probability of seeing both reward values in one group is

$$\Pr(\text{mixed rewards}) = 1 - p_b^G - (1 - p_b)^G. \tag{8}$$

A mixed group has nonzero individual reward advantages. Its summed gradient can still cancel, and a finite update can fail to improve the local objective or the original terminal objective. An outcome observed once does not establish that future batches have a favorable p_b. Fresh sampling, start adjustment, and initial-state full attempts are therefore necessary operational checks, rather than consequences of this identity.

The zero-advantage statement concerns only environment reward. KL regularization may generate an update even when rewards coincide; that update constrains policy drift rather than discriminating successful task behavior. The complete regularized objective is in Appendix C.2.

Incomplete observations and dependence. An unknown checker result is not a Bernoulli failure. The implementation withholds an RL update for a group whose rewards cannot be resolved under its fixed protocol. Equation 7 applies to complete-observation sampling; it does not imply unbiasedness after outcome-dependent group selection. We record missingness and skipped groups, and restrict any unbiased-gradient interpretation to complete feedback or an independently justified missingness condition. Sharing an active service, changing memory between group members, selecting only successful starts, or updating parameters before the group is complete likewise invalidates the stated sampling argument.

B.2 Compatible Sharing and Estimation Uncertainty

Fix an operation or skill, its executor version, observable preconditions, initialization rule, and horizon. Let $Y_\ell \in \{0,1\}$ indicate whether a fresh execution achieves its specified effect. Suppose n trials are independent and have the same effect probability p. With $\widehat{p} = n^{-1}\sum_{\ell=1}^{n} Y_\ell$, Hoeffding's inequality gives, with probability at least $1-\delta$,

$$|\widehat{p} - p| \le \sqrt{\frac{\log(2/\delta)}{2n}}. \tag{9}$$

This standard bound (Hoeffding, 1963) explains why genuinely compatible evidence from multiple tasks can reduce uncertainty about a local effect. The common quantity is the effect probability under the specified execution conditions, not the final success probability of the source tasks.

To expose the cost of imperfect transfer, let source j contribute a fixed number n_j of independent trials with effect probability p_j, let $n = \sum_j n_j$, and let $w_j = n_j/n$. For a target effect probability p_{tar}, concentration around $\overline{p} = \sum_j w_j p_j$ and the triangle inequality yield

$$|\widehat{p} - p_{\mathrm{tar}}| \le \sqrt{\frac{\log(2/\delta)}{2n}} + \sum_j w_j |p_j - p_{\mathrm{tar}}| \tag{10}$$

with probability at least $1-\delta$. The first term decreases with independent sample count; the second expresses source–target mismatch. Adding data can reduce the former while increasing the latter.

Neither p_j nor the mismatch term is known to the agent. Matching visible software versions, input formats, object types, sessions, or permissions is an operational filter, not proof that the conditional dynamics agree. Unobserved configuration and history can still matter. The method consequently preserves scope and counterexamples and verifies retrieved operations in the target instance. The negative-transfer protocol in Appendix C.4 directly tests these safeguards.

Candidate-construction observations are not independent validation evidence for an adaptively chosen candidate. Fixed validation batches are collected after candidate specification. Repeated queries within one unchanged session, correlated trajectories, and adaptive stopping do not supply the independent sample count assumed above. The bounds also concern correctness under a fixed, semantically valid checker; they do not bound systematic checker error. These restrictions motivate the separation of checker-quality testing from execution repeatability in Appendices C.4 and C.2.

B.3 Conditions for Continuous Skill Composition

Consider a particular executable route with L skills within a total task horizon. Let A_0 be the event that an execution from the original initial distribution reaches the first skill's valid context. For $\ell = 1, \dots, L$, let A_ℓ mean that skill ℓ succeeds and, when another skill follows, leaves its required context valid. The final event includes the native terminal check. Thus $\cap_{\ell=0}^{L} A_\ell$ implies complete task success. The assumptions concern the boundary-state distributions induced by this actual continuous execution, including history and environmental randomness.

If $\Pr(A_0) \ge p_0$ and $\Pr(A_\ell \mid A_0, \dots, A_{\ell-1}) \ge p_\ell$ for every defined conditional event, the chain rule gives

$$\begin{aligned} \Pr(\text{complete success}) &\ge \Pr(A_0) \prod_{\ell=1}^{L} \Pr(A_\ell \mid A_0, \dots, A_{\ell-1}) \\ &\ge p_0 \prod_{\ell=1}^{L} p_\ell. \end{aligned} \tag{11}$$

This is a probability identity followed by conditional lower bounds, not an independence theorem. If a prefix event has zero probability, the nontrivial lower-bound assumptions fail; the resulting zero lower bound provides no reachability information.

Measured success rates from separately prepared checkpoints cannot simply be substituted for the conditional probabilities in Equation 11. A prefix can change input files, privileges, process state, session validity, or the object being accessed, thereby changing the next skill's distribution. Conditions in a proposed AND set can invalidate one another. Real prefix re-execution and current-context checks address these issues operationally, and continuous full attempts measure whether a route actually composes. Improving one local skill need not improve complete performance if reaching it or preserving its successor context remains unlikely. No complete route is assumed to be available during the strict zero-success phase.

B.4 From Demonstrated Histories to Autonomous Execution

SFT minimizes an action loss on histories supplied by the success archive. Autonomous actions determine subsequent observations, so small deviations can expose histories absent from that archive. This sequential distribution-shift issue is a standard motivation for collecting data under the learner's own behavior (Ross et al., 2011). Bi-PGSD uses fresh self-generated interactions and terminal feedback, rather than assuming an expert can relabel arbitrary learner states.

The withdrawal curriculum changes both the starting-context distribution and the available guidance. Fresh evaluation at every level is needed because successful suffix execution does not establish autonomous prefix generation. Script repeatability, action likelihood on demonstrations, prefix reach rates, and descriptive overlap with student-visited contexts can diagnose different parts of this gap. They remain auxiliary analyses: neither a coverage loss nor a new coverage-based objective is introduced. The relevant empirical endpoints are initial-state success, retention after subsequent training, and frozen held-out performance under explicitly stated scaffolding conditions.

C Compact Implementation and Experimental Controls

C.1 Task lifecycle, information access, and evidence

A run fixes the benchmark release, environment/build, native task input, tool schema, authorization scope, initialization procedure, and terminal verifier. Hidden reference answers, private evaluator state, writeups, and manual decompositions are excluded from strict zero-success learning; self-generated failed experience may be retained with source provenance. Local predicates use three-valued checks $V_p(h) \in \{0, 1, \perp\}$, where $\perp$ denotes insufficient evidence rather than failure. Every reusable record binds its condition, action, outcome, checker version, executor/policy version, object/session context, supporting or conflicting executions, and cost.

C.2 Candidate validation, goal regression, and local learning

A backward condition is a proposal, not evidence. Bi-PGSD first checks reachability and applicability in the current environment, then turns supported outcomes into the local goals defined in Equation 3. Starts are established by reset and real-prefix replay (or a validated consistent checkpoint), and all return/checking operations are charged. Fresh trajectories provide the RLOO updates of Equation 4; unknown local outcomes are excluded from the corresponding policy-gradient group and logged rather than silently coded as failures. The fixed terminal verifier remains the only criterion for complete-task success.

C.3 Scheduling, success admission, and consolidation

A progress scheduler allocates effort using local improvement and execution cost while reserving original-state attempts. Stalled tasks retain unmet conditions; experience acquired elsewhere is reused only after target-side validation. A complete path enters the positive training set only after independent clean-instance reproduction. Consolidation then trains from admitted successes and progressively withdraws retrieval and intermediate guidance, keeping discovery, script reproduction, and scaffold-free policy generation as distinct endpoints.

C.4 Mechanism, transfer, and autonomy controls

The reported controls vary one of four factors at a time: experience representation, goal direction, environment validation, or post-success scaffolding. Cross-task comparisons charge source work and target-side revalidation to the same total budget; mismatched-condition arms measure negative transfer rather than discarding it. Checker tests use independent observation–predicate evaluations, and environment interventions use a separate complete-task endpoint. Post-success studies distinguish replayability from learnability, compare immediate and progressive scaffold withdrawal, and evaluate frozen held-out capability with no test-time updating. Full numerical outcomes, denominators, costs, and model/environment strata are reported in Appendices D–H.

C.5 Minimal run manifest

A reproducible run needs only the configuration fields below in addition to the submitted core code and the cited benchmark assets; the paper does not repeat controller-level implementation prose.

Table 3: Minimal configuration required to reproduce a reported run.

Field	Required value
Task identity	Benchmark release, task/instance ID, environment image or build, native input level, and terminal verifier.
Model/update state	Base model, policy/checker/predictor versions, optimizer configuration, decoding, and frozen/update boundaries.
Information boundary	Permitted task inputs, retrieval sources, source-task eligibility, and evaluation-only artifacts.
Budgets	Episode/action/token/verifier limits plus replay, validation, source-task, and training accounting.
Local learning	Candidate limits, checker version, local horizon, group size, KL coefficient, and start/return rule.
Randomness	Screening/evaluation seeds, instance-reset randomness, and any fixed candidate or intervention cohort.
Success handling	First-success event, independent reproduction rule, positive-set admission, and scaffold-withdrawal schedule.

D Results, Cost, and Configuration Details

Experimental-result provenance. The numerical results reported in this work are measured outcomes from the experimental configurations, task manifests, and budgets specified in the paper. Model names denote the evaluated checkpoints or API configurations used for those runs. Because the task manifests and resource accounting do not necessarily coincide with each benchmark's official test protocol, we report these measurements only under the stated protocol; published literature results are listed separately under their original protocols.

D.1 Splits and Configuration Registry

Training, development, online discovery, and frozen holdout use disjoint manifests at the project/template/family level. Each base policy is screened independently; Z reports first success over strict zero-success pools, whereas A uses target exposures as its denominator. Table 4 gives the eligible counts and common intersections; Table 5 defines which candidate strata permit parameter updates. These tables define the comparison population without implying a cross-model capability ranking.

Table 4: Eligible task counts after finite screening. Each environment has a task universe of 160, not an official test set. The final row is the common zero-success intersection across all candidate models.

Model	Cy	NYU	IC	CVE	CG	V8	SM
Qwen3.5-9B	140	147	44	140	149	147	146
Qwen3.5-27B	129	133	41	127	139	139	138
Qwen3.8-27B	126	127	29	132	132	139	145
GLM-4.7-Flash	140	143	39	136	134	137	142
Gemma 4 26B-A4B	133	137	43	124	141	138	134
gpt-oss-20b	132	138	51	130	142	137	146
GPT-6 Astra	115	115	21	114	125	117	126
Claude Opus 5.5	109	112	17	112	122	122	122
Gemini 3.8 Flash	126	122	26	120	128	131	134
DeepSeek-V4.1-Flash	120	132	31	101	132	125	135
GLM-5.3-Flash	120	138	24	122	129	129	138
Kimi K3	116	125	28	109	131	120	133
Common intersection	30	24	0	24	41	36	38

Table 5: Model configurations and update boundaries used in the reported experiments.

Candidate	Evaluation configuration
Qwen3.5-9B	Full local learning candidate
Qwen3.5-27B	Full local learning candidate
Qwen3.8-27B	Full local learning candidate
GLM-4.7-Flash	Full local learning candidate
Gemma 4 26B-A4B	Full local learning candidate
gpt-oss-20b	Full local learning candidate
GPT-6 Astra	Frozen policy; external updates only
Claude Opus 5.5	Frozen policy; external updates only
Gemini 3.8 Flash	Frozen policy; external updates only
DeepSeek-V4.1-Flash	Frozen policy; external updates only
GLM-5.3-Flash	Frozen policy; external updates only
Kimi K3	Frozen policy; external updates only

D.2 End-to-End Cost and Statistical Analysis

All online work is charged to seven mutually exclusive accounts: policy inference, experience extraction, backward guidance, environment verification, prefix replay, parameter training, and source-task work. The common budget is a cap rather than guaranteed consumption; shared screening and manual audit remain outside this online axis. Total cost is

$$C_{\text{total}} = \sum_{a \in \mathcal{A}} \big(r_{\text{in}} T_{a,\text{in}} + r_{\text{out}} T_{a,\text{out}} + r_{\text{GPU}} H_{a,\text{GPU}} + r_{\text{CPU}} H_{a,\text{CPU}} \big), \tag{12}$$

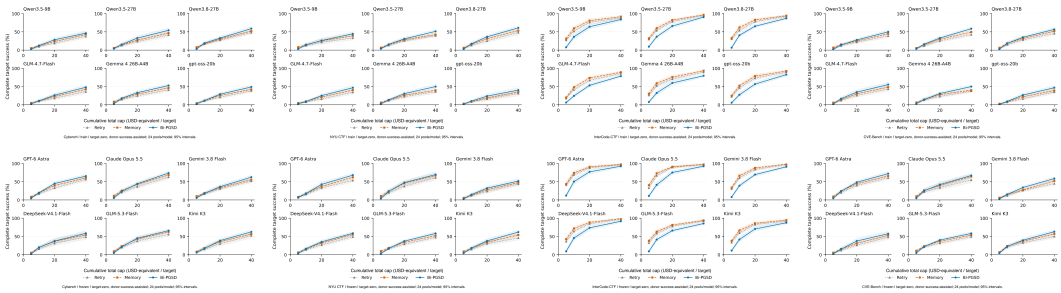
Task pools/training seeds are the independent broad-coverage units. Reported intervals use task-clustered paired bootstrap resampling within model/environment strata; unsuccessful first-success events are right-censored at the common cap. Monetary matching does not imply matched tokens, FLOPs, interactions, or wall-clock time.

D.3 Published Results under Their Original Protocols

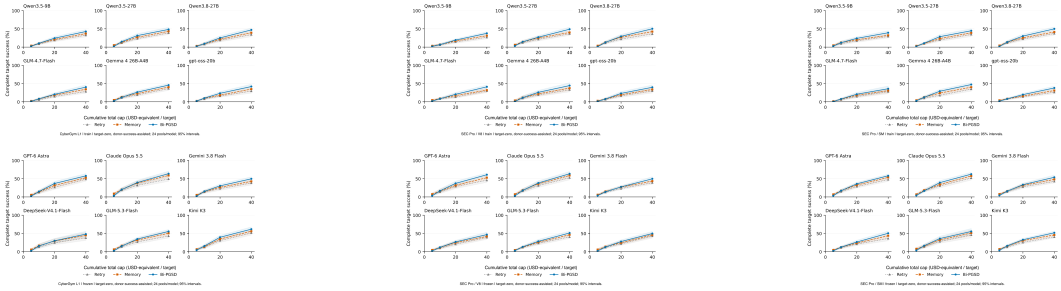
Published Cyber-Zero, CyberGym, and SEC-bench Pro numbers are retained only under their original agents, data access, task versions, and budgets. They are context rather than matched baselines: writeup supervision, task collections, model checkpoints, and cost definitions differ from the present evaluation, and unreported quantities are left missing rather than reconstructed.

Table 6: Published results under their original protocols, not matched baselines. A dash denotes an unreported item or one that cannot be verified from published aggregates; per-task records are not reconstructed.

Agent / model	Benchmark	Protocol	Score (%)	Cost (USD)
zero-shot / Qwen3-32B (Zhuo et al., 2026)	InterCode-CTF	40 turns	60.0	–
Cyber-Zero SFT / Qwen3-32B (Zhuo et al., 2026)	InterCode-CTF	40 turns	82.4	–
zero-shot / Qwen3-32B (Zhuo et al., 2026)	NYU CTF	40 turns	4.7	–
Cyber-Zero SFT / Qwen3-32B (Zhuo et al., 2026)	NYU CTF	40 turns	13.5	–
zero-shot / Qwen3-32B (Zhuo et al., 2026)	Cybench	40 turns	5.0	–
Cyber-Zero SFT / Qwen3-32B (Zhuo et al., 2026)	Cybench	40 turns	17.5	–
OpenHands / Claude-3.7-Sonnet (Wang et al., 2026)	CyberGym L1	L1; 100 iterations; approx. USD 2 average	11.9	–
OpenHands / GPT-4.1 (Wang et al., 2026)	CyberGym L1	L1; 100 iterations; approx. USD 2 average	9.4	–
OpenHands / DeepSeek-V3 (Wang et al., 2026)	CyberGym L1	L1; 100 iterations; approx. USD 2 average	3.6	–
Codex / GPT-5.4 (Lee et al., 2026)	SEC / V8	5400 s	32.0	9.97
OpenCode / Kimi-K2.6 (Lee et al., 2026)	SEC / V8	5400 s	11.7	6.88
ClaudeCode / Opus 4.6 (Lee et al., 2026)	SEC / Spider-Monkey	5400 s	38.8	15.22



(a) Cybench (b) NYU CTF (c) InterCode-CTF (d) CVE-Bench



(e) CyberGym L1 (f) SEC Pro / V8 (g) SEC Pro / SpiderMonkey

Figure 5: Combined A-condition curves. Each panel stacks the trainable-policy and frozen-policy candidates; all environments use the same budget grid and pool-bootstrap convention.

E Experience Representations, Checkers, Conditions, and Rewards

E.1 From Failed Fragments to Checkable Local Learning Tasks

E06–E12 test whether failed interaction becomes useful only after it is distinguishable, evidence-linked, and goal-relevant. On Cybench, verified local learning raises held-out local completion from 16.8% to 34.0%, and evidence-filtered failed fragments outperform discarding, random fragments, or globally downweighting failed actions. Conditional representations outperform raw retrieval, reflection, and structure without evidence; blocking knowledge before use produces a clear drop, while later restoration only partly recovers performance. Goal-directed selection remains robust when novelty is distracted, and scope-checked links outperform random or swapped links. In the forward/backward factorial, both directions contribute, but the Cybench interaction is 2.5 percentage points with 95% interval [-0.1, 5.2], supporting complementary rather than superadditive effects. Full cells are in Table 8.

E.2 Environment Interventions and Checker Quality

E13 intervenes directly on candidate conditions: satisfying rather than sham-modifying a condition changes Cybench terminal success by 10.8 percentage points, with blocked and unrelated-condition arms as controls. E14 independently evaluates observation–predicate judgments. The updated checker reduces false positives from 4.4% to 2.3% and false negatives from 11.5% to 6.0%, while coverage on adequately evidenced samples rises from 85.5% to 93.9%; it still accepts 4.3% of insufficient-evidence cases. Figure 6 and Table 7 report the terminal and checker endpoints separately.

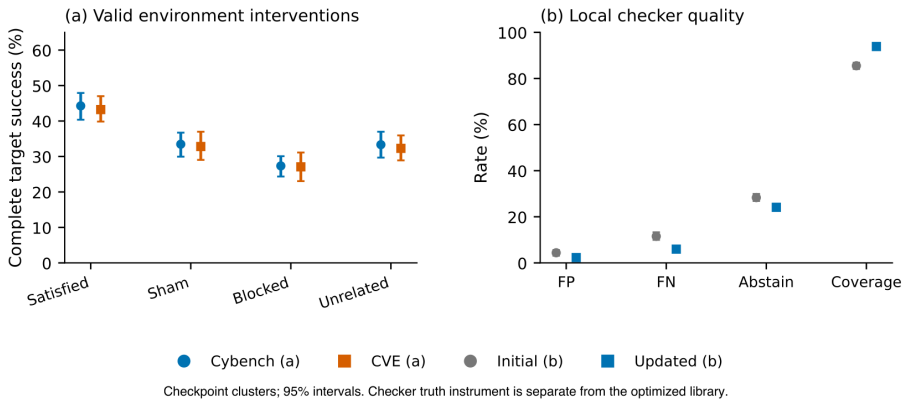


Figure 6: Left: complete success after legal environment-condition interventions. Right: local-checker metrics on independent observation–predicate tests; these are different endpoints. Environment interventions use 16 source checkpoints per model, and checker tests use 16 test clusters. Error bars are 95% cluster intervals. Ground-truth instrumentation is isolated from the updatable checker library; denominators are specified in Table 7.

Table 7: Independent observation–predicate tests. False-positive rates use true negatives; false-negative rates use true positives. Coverage is the fraction of sufficiently evidenced samples receiving a decision. Insufficient-evidence samples are reported separately.

Metric (%)	Initial checker	Updated checker
False positive	4.4 [3.1, 5.6]	2.3 [1.4, 3.3]
False negative	11.5 [10.0, 13.3]	6.0 [4.9, 7.2]
Abstention, all items	28.3 [26.8, 29.9]	24.1 [22.9, 25.6]
Coverage, adequate evidence	85.5 [84.1, 86.8]	93.9 [92.5, 95.1]
Accept, inadequate evidence	8.7 [7.0, 10.4]	4.3 [2.7, 6.0]

E.3 Module Updates, Rewards, and Learning Algorithms

E15–E17 separate module updates from policy learning. Freezing either the checker or guidance lowers terminal success, and freezing both lowers it further. Environment-anchored local reward outperforms terminal reward only and calibrated model progress reward. Filtered local SFT already captures part of the gain, while adding local RL recovers the full Bi-PGSD level; on Cybench, the local-RL versus filtered-SFT difference is 3.3 percentage points. Table 8 gives the full stratified cells and keeps E06 local completion separate from terminal success.

Table 8: Stratified mechanism results with two fixed model families per environment. Cells report success rates and intervals (%). E06 measures local-goal completion; all other rows measure terminal completion. Distinct endpoints are not pooled.

ID	Controlled condition	Cybench	CVE-Bench
E06	Zero terminal rewards	16.8 [14.1, 19.5]	16.0 [13.5, 18.6]
	Verified local learning	34.0 [29.6, 38.5]	28.9 [25.9, 31.9]
E07	Discard failures	27.0 [23.4, 30.5]	22.7 [19.8, 25.4]
	Evidence-filtered local fragments	38.2 [33.9, 42.4]	29.6 [26.2, 33.2]
	Random fragments	28.6 [24.7, 32.4]	24.1 [21.4, 26.8]
	Globally downweight failure actions	24.2 [20.6, 27.6]	20.2 [17.7, 22.8]
E08	Bi-PGSD	44.7 [41.8, 47.4]	38.3 [34.6, 41.8]
	Strong-summary retrieval	38.2 [34.8, 41.5]	32.6 [29.2, 35.8]
	Raw episode retrieval	32.3 [29.3, 35.4]	27.3 [24.2, 30.5]
	Free-text reflection	34.1 [31.1, 37.1]	28.8 [25.7, 32.0]
	Structure without evidence	34.5 [31.2, 37.6]	29.8 [26.7, 32.9]

Continued

ID	Controlled condition	Cybench	CVE-Bench
E09	Provided	40.5 [37.1, 43.8]	37.2 [32.7, 41.5]
	Blocked before use	28.4 [24.7, 32.0]	25.7 [21.9, 29.4]
	Unrelated replacement	29.3 [25.8, 32.9]	25.3 [21.5, 28.9]
	Restored at B/2	35.8 [32.4, 39.1]	32.9 [28.6, 37.1]
E10	low distractors / novelty	34.1 [30.9, 37.4]	32.2 [28.8, 35.8]
	low distractors / goal	33.9 [30.7, 37.2]	32.3 [28.9, 35.8]
	high distractors / novelty	25.0 [21.6, 28.3]	22.4 [19.7, 25.4]
	high distractors / goal	34.0 [30.9, 37.0]	31.4 [27.9, 35.0]
E11	Bi-PGSD	44.7 [41.8, 47.4]	38.3 [34.6, 41.8]
	F- B-	23.8 [20.8, 26.8]	19.8 [16.5, 22.9]
	F+ B-	34.4 [31.4, 37.4]	28.6 [25.3, 31.9]
	F- B+	31.6 [29.0, 34.2]	27.3 [24.2, 30.6]
E12	Scope-checked matching	39.7 [35.9, 43.4]	39.2 [34.9, 43.6]
	Semantic matching	36.7 [32.9, 40.6]	37.2 [33.3, 41.1]
	Type-matched random links	28.4 [24.6, 32.0]	27.7 [24.2, 30.9]
	Swapped target links	28.0 [24.5, 31.6]	25.9 [22.1, 29.3]
	Correct links restored	37.2 [33.1, 41.4]	37.1 [32.8, 41.1]
E13	Condition satisfied	44.3 [40.4, 47.9]	43.2 [39.8, 47.0]
	Sham intervention	33.5 [29.9, 36.7]	32.8 [29.0, 37.0]
	Condition blocked	27.3 [24.3, 30.1]	27.1 [23.0, 31.1]
	Unrelated condition satisfied	33.3 [29.7, 37.0]	32.3 [28.9, 35.9]
E15	Bi-PGSD	44.7 [41.8, 47.4]	38.3 [34.6, 41.8]
	Frozen local checker	38.0 [35.2, 40.8]	31.9 [28.6, 35.4]
	Frozen guidance	38.3 [35.3, 41.1]	32.4 [29.4, 35.7]
	Checker + guidance fixed	32.3 [29.6, 35.2]	27.2 [23.8, 30.5]
E16	Terminal reward only	26.0 [23.0, 29.0]	24.1 [21.2, 27.2]
	Environment-anchored local reward	40.0 [35.3, 44.5]	36.8 [33.1, 40.9]
	Calibrated model progress reward	35.4 [31.1, 39.2]	32.6 [28.9, 36.2]
	Verified reward + shaping	38.8 [34.4, 43.2]	37.5 [33.6, 41.5]
E17	Bi-PGSD	44.7 [41.8, 47.4]	38.3 [34.6, 41.8]
	Filtered local SFT	41.4 [38.3, 44.8]	35.5 [32.3, 38.9]
	No local policy updates	34.1 [31.2, 37.1]	28.9 [25.8, 32.3]
	SFT then local RL	44.7 [41.7, 47.8]	38.2 [34.4, 42.1]

F Sharing Granularity, Transfer Channels, Scheduling, and Negative Transfer

F.1 What to Share and through Which Channel

E18–E20 vary source content, granularity, and transfer channel. Verified local failure experience improves over withholding or unrelated fragments; complete or ordered successful traces outperform shuffled or random fragments; and both external knowledge and parameter transfer contribute when branched from the same source checkpoint. The Cybench interaction interval between the two sharing factors, [-0.8, 4.8], includes zero, so the gains do not require a superadditive interpretation. Table 9 contains the full stratified comparisons.

F.2 Can Task Switching Repay Source Costs?

E21–E22 compare task allocation under the same total budget. On Cybench, a related-source detour improves terminal success over direct continuation by 5.6 percentage points, whereas unrelated detours and transfer-blocked returns do not recover the same gain. The interval comparing the complex stagnation scheduler with round-robin, [-0.7, 3.1], includes zero. Thus useful cross-task detours do not imply that complex scheduling itself is necessary; full cells are in Table 9.

F.3 The Costs of Scope Checks and Revalidation

E23 isolates transfer safeguards. Removing scope checks, target-side revalidation, or conflict handling increases negative transfer, but verification is not free: under low-risk conditions, stronger revalidation changes Cybench terminal success by -3.1 percentage points. This gives a cost-sensitive boundary rather than a monotonic "more checking is better" rule. Table 9 reports the corresponding conditions.

Table 9: Stratified mechanism results with two fixed model families per environment. Cells report terminal success rates and intervals (%).

ID	Controlled condition	Cybench	CVE-Bench
E18	Bi-PGSD	44.7 [41.8, 47.4]	38.3 [34.6, 41.8]
	S- L-	25.4 [22.7, 28.1]	21.4 [18.5, 24.2]
	S+ L-	32.9 [29.9, 35.9]	28.6 [25.4, 31.8]
	S- L+	35.2 [31.9, 38.4]	29.6 [26.3, 33.1]
E19	Bi-PGSD	44.7 [41.8, 47.4]	38.3 [34.6, 41.8]
	Whole failed traces shared	36.1 [33.1, 38.8]	30.3 [27.0, 33.5]
	Full successful trajectories	45.1 [42.2, 47.9]	38.9 [35.2, 42.8]
	Ordered compressed trajectories	44.8 [41.9, 47.7]	39.1 [35.3, 42.8]
	Unordered local effects	39.1 [36.2, 42.2]	34.1 [31.0, 37.4]
	Evidence-linked failed summaries	42.6 [39.7, 45.4]	38.0 [34.2, 41.9]
	Random failed fragments	30.6 [27.7, 33.7]	25.8 [22.9, 28.8]
E20	Old parameters / old knowledge	25.8 [22.4, 28.9]	25.1 [21.6, 28.6]
	Old parameters / new knowledge	35.5 [32.3, 38.7]	33.6 [30.1, 37.1]
	New parameters / old knowledge	33.2 [30.1, 36.5]	32.9 [29.7, 36.2]
	New parameters / new knowledge	42.3 [39.1, 45.4]	40.2 [36.2, 44.1]
E21	Bi-PGSD	44.7 [41.8, 47.4]	38.3 [34.6, 41.8]
	Round-robin scheduling	43.4 [40.4, 46.1]	37.8 [34.0, 41.4]
	Sequential allocation	36.3 [33.3, 39.3]	31.6 [28.4, 35.2]
	Random switching	40.1 [37.2, 43.0]	34.1 [30.2, 37.8]
	Learning-progress scheduler	43.9 [40.9, 46.9]	37.5 [34.1, 41.3]
	Stagnation / random return	42.3 [39.3, 45.4]	36.8 [32.9, 40.5]
E22	Continue target	34.8 [31.2, 38.3]	30.2 [27.2, 33.6]
	Related detour	40.4 [36.7, 44.3]	33.6 [29.9, 37.2]
	Unrelated detour	30.2 [27.0, 33.7]	25.1 [22.4, 28.3]
	Detour, all channels blocked	31.4 [27.9, 34.8]	25.7 [22.8, 28.6]
E23	Scope + target revalidation	40.6 [37.2, 44.3]	36.8 [31.9, 42.1]
	No scope restriction	31.6 [28.4, 35.2]	29.3 [24.7, 34.1]
	No target revalidation	35.5 [32.7, 38.7]	33.2 [28.4, 38.5]
	Overwrite conflicts	35.5 [32.0, 39.2]	32.7 [27.9, 37.9]
	Low-risk / no revalidation	44.3 [40.9, 47.7]	39.8 [35.0, 44.5]
	Low-risk / strong revalidation	41.1 [37.8, 44.7]	38.3 [33.6, 43.8]

G Success-Trajectory Learning, Withdrawal, Retention, and Cross-Cyber Transfer

G.1 Reliable Seeds versus Learnable Demonstrations

E24–E26 separate discovery, replayability, and learnability. On Cybench, a discovered script replays successfully after reset in 77.6% of cases, whereas scaffold-free model regeneration succeeds in only 25.9%. Confirming a seed before amplification outperforms immediate amplification, and student-regenerated or equal-cost newly discovered successes perform comparably to training on the original successful traces. Table 11 reports all conditions and intervals.

G.2 Parameter–Scaffolding Combinations and Progressive Withdrawal

E04 and E27–E28 test whether success survives removal of retrieval, intermediate prompts, preset starts, and prefix assistance. Progressive withdrawal improves scaffold-free performance over success-only SFT across the model/environment grid in Table 10. In the Cybench family holdout, gradual removal reaches 20.4%, and the later retention checkpoint remains at 18.7%. Same-family new-instance and family-held-out tests are frozen, with no test-time parameter or memory updates.

Table 10: Complete model and environment coverage for frozen scaffold-free evaluation. Each cell gives same-family new-instance/family-held-out success (%), with distinct denominators and no test-time updates.

Environment	Model	Success SFT	Gradual removal
Cybench	Qwen3.5-9B	20.1/16.0	22.9/18.1
	Qwen3.5-27B	22.2/22.9	26.7/25.0
	Qwen3.8-27B	24.3/21.2	27.4/22.2
	GLM-4.7-Flash	18.4/14.2	20.5/17.4
	Gemma 4 26B-A4B	23.3/18.8	25.7/20.5
	gpt-oss-20b	23.6/17.4	27.1/19.4
NYU CTF	Qwen3.5-9B	23.6/16.7	26.0/19.8
	Qwen3.5-27B	23.3/23.6	26.4/26.7
	Qwen3.8-27B	25.0/20.8	28.5/22.6
	GLM-4.7-Flash	19.8/8.7	21.9/10.8
	Gemma 4 26B-A4B	21.9/19.1	24.0/20.1
	gpt-oss-20b	20.5/14.9	22.2/17.0
InterCode-CTF	Qwen3.5-9B	43.4/38.2	45.8/40.3
	Qwen3.5-27B	46.2/37.2	47.6/41.7
	Qwen3.8-27B	49.7/36.8	52.8/40.6
	GLM-4.7-Flash	45.5/38.2	49.3/41.3
	Gemma 4 26B-A4B	52.8/37.5	54.5/42.4
	gpt-oss-20b	54.2/32.6	56.6/35.4
CVE-Bench	Qwen3.5-9B	29.9/16.7	32.6/18.1
	Qwen3.5-27B	26.7/20.8	31.6/22.9
	Qwen3.8-27B	25.7/25.0	27.8/27.4
	GLM-4.7-Flash	20.8/18.1	25.0/20.5
	Gemma 4 26B-A4B	26.4/13.2	28.8/15.6
	gpt-oss-20b	25.0/18.1	28.1/20.1
CyberGym L1	Qwen3.5-9B	18.8/15.3	19.8/18.4
	Qwen3.5-27B	23.6/14.9	26.0/16.3
	Qwen3.8-27B	24.3/18.4	28.1/22.2
	GLM-4.7-Flash	20.1/18.4	23.3/22.9
	Gemma 4 26B-A4B	20.8/17.0	22.6/19.1
	gpt-oss-20b	20.1/21.2	24.0/24.0
SEC Pro / V8	Qwen3.5-9B	19.4/10.8	22.2/13.2
	Qwen3.5-27B	25.0/16.0	28.8/19.1
	Qwen3.8-27B	25.0/15.6	28.8/17.4
	GLM-4.7-Flash	25.7/14.2	27.4/16.0
	Gemma 4 26B-A4B	24.7/15.6	28.1/18.1
	gpt-oss-20b	21.2/16.7	22.2/18.8
SEC Pro / SM	Qwen3.5-9B	21.5/14.6	23.3/16.7
	Qwen3.5-27B	22.9/16.7	26.0/19.8
	Qwen3.8-27B	21.9/18.8	26.0/20.5
	GLM-4.7-Flash	23.3/12.5	26.0/13.5
	Gemma 4 26B-A4B	21.2/10.8	25.0/12.2
	gpt-oss-20b	20.5/15.6	25.0/18.8

G.3 Cross-Cyber Environment Transfer

E29 tests transfer to a different cyber environment. Parameter-only and external-experience-only transfer both improve over no transfer, while combining both channels with target

adaptation gives the highest terminal success: 29.2% on Cybench and 26.8% on CVE-Bench in Table 11. This endpoint permits target-side adaptation and is reported separately from the frozen scaffold-free holdout in E28.

Table 11: Stratified mechanism results with two fixed model families per environment. Cells report terminal success rates and intervals (%).

ID	Controlled condition	Cybench	CVE-Bench
E24	Script replay after reset	77.6 [74.1, 81.0]	76.2 [73.2, 79.3]
	Model regeneration, no scaffold	25.9 [23.0, 29.0]	26.0 [21.9, 29.9]
	Confirm before amplification	38.9 [34.8, 42.8]	36.3 [32.0, 40.5]
	Amplify immediately	36.8 [33.2, 40.5]	34.9 [30.1, 39.5]
	No amplification	27.7 [24.5, 31.4]	26.7 [22.4, 30.9]
E25	Student-discovered successful traces	38.3 [34.1, 42.6]	36.3 [32.9, 40.1]
	External-model successful traces	36.1 [32.3, 40.0]	35.4 [32.0, 38.9]
	Composed then continuously validated	37.2 [33.2, 41.4]	35.2 [31.2, 38.7]
E26	Original successful traces	37.0 [33.1, 41.3]	33.2 [29.8, 36.6]
	Student regeneration	39.8 [36.1, 43.6]	35.8 [32.0, 39.3]
	Equal-cost new successes	39.8 [35.9, 43.9]	35.5 [31.6, 39.3]
E29	No cross-environment transfer	23.3 [20.4, 26.4]	20.7 [18.0, 23.7]
	Parameter transfer only	26.6 [23.3, 29.8]	24.5 [21.0, 28.3]
	External experience only	26.2 [22.9, 29.4]	23.7 [20.6, 27.1]
	All channels + target adaptation	29.2 [26.0, 32.8]	26.8 [23.7, 30.6]

H Theoretical Boundaries and Exploratory Phenomena

H.1 When Bidirectional Discovery Should Lose Its Advantage

E30–E32 vary depth/branching, candidate quality, observability/reset access, and validation repetition under fixed budgets. Bidirectional exploration remains stronger than forward-only search when candidate quality and intermediate evidence are informative, but its advantage disappears under hidden feedback or unavailable resets. Repetition is nonmonotonic: moderate repeats help under independent checker noise, whereas excessive repetition spends budget and systematic error is not removed by repeating the same checker. Figure 7 and Table 12 report the full sweep.



Figure 7: Boundary conditions in authorized toy cyber designs. (a) Depth and candidate quality at fixed broad branching: blue/orange denotes bidirectional/forward-only exploration, and solid/dashed lines denote high/low quality. (b) Blue/orange denotes bidirectional/retry conditions as observability and reset access decrease. (c) Fixed-budget repetition sweep under independent/systematic errors. Curves illustrate boundary conditions; full factors and intervals are in Table 12. Hidden distances are not used for optimization.

Table 12: Stratified mechanism results with two fixed model families per environment. Cells report terminal success rates and intervals (%).

ID	Controlled condition	Cybench	CVE-Bench
E30	d=2, b=2, high, forward	33.6 [29.6, 37.2]	29.8 [26.6, 33.2]
	d=2, b=2, high, bidirectional	43.5 [39.3, 47.8]	42.7 [38.2, 46.7]
	d=2, b=2, low, forward	33.5 [29.6, 37.5]	30.7 [27.2, 34.2]
	d=2, b=2, low, bidirectional	31.6 [27.9, 35.0]	28.6 [25.4, 31.8]
	d=2, b=8, high, forward	29.7 [26.0, 33.3]	26.7 [23.6, 30.1]
	d=2, b=8, high, bidirectional	39.8 [35.8, 43.9]	37.6 [33.3, 41.8]
	d=2, b=8, low, forward	30.6 [27.1, 34.2]	26.0 [22.8, 29.3]
	d=2, b=8, low, bidirectional	27.3 [24.0, 30.9]	24.5 [21.2, 28.0]
	d=4, b=2, high, forward	28.1 [24.5, 32.2]	25.8 [22.4, 29.2]
	d=4, b=2, high, bidirectional	39.1 [34.9, 43.4]	35.9 [31.8, 40.2]
	d=4, b=2, low, forward	28.5 [24.9, 32.2]	25.4 [22.1, 28.8]
	d=4, b=2, low, bidirectional	25.5 [22.3, 28.8]	23.4 [20.2, 27.0]
	d=4, b=8, high, forward	24.6 [21.6, 27.2]	22.5 [19.5, 26.0]
	d=4, b=8, high, bidirectional	35.0 [31.4, 38.8]	31.5 [28.0, 35.3]
	d=4, b=8, low, forward	25.0 [21.6, 28.5]	22.9 [20.0, 25.9]
	d=4, b=8, low, bidirectional	22.0 [19.5, 24.5]	20.3 [17.4, 23.3]
	d=8, b=2, high, forward	19.5 [17.3, 21.9]	18.4 [15.9, 20.8]
	d=8, b=2, high, bidirectional	29.3 [25.7, 32.8]	25.4 [22.1, 29.0]
	d=8, b=2, low, forward	19.1 [16.7, 21.5]	18.1 [15.1, 21.2]
	d=8, b=2, low, bidirectional	18.2 [16.0, 20.4]	16.3 [13.8, 18.9]
	d=8, b=8, high, forward	16.7 [14.3, 18.8]	14.8 [12.1, 17.4]
	d=8, b=8, high, bidirectional	24.7 [21.7, 27.7]	23.0 [19.9, 26.3]
	d=8, b=8, low, forward	16.0 [14.1, 18.2]	15.2 [13.2, 17.6]
	d=8, b=8, low, bidirectional	15.5 [13.4, 17.6]	14.1 [11.6, 16.5]
E31	observed, retry	40.2 [37.1, 43.9]	32.9 [29.6, 36.7]
	observed, bidirectional	50.3 [46.6, 54.2]	41.4 [37.9, 45.1]
	hidden, retry	11.5 [8.9, 14.3]	8.6 [6.5, 10.7]
	hidden, bidirectional	11.5 [8.6, 14.7]	8.1 [6.1, 10.2]
	no_reset, retry	25.5 [21.9, 29.2]	19.3 [16.9, 21.7]
	no_reset, bidirectional	25.7 [22.1, 29.3]	19.1 [16.7, 21.9]
E32	0 repeats / independent error	31.4 [28.0, 34.8]	29.2 [25.8, 32.7]
	0 repeats / systematic error	29.6 [26.0, 33.1]	28.3 [25.0, 31.5]
	1 repeats / independent error	34.9 [32.2, 37.8]	31.9 [28.4, 35.3]
	1 repeats / systematic error	27.2 [23.8, 30.5]	26.8 [23.6, 29.9]
	2 repeats / independent error	37.9 [34.6, 41.3]	34.0 [30.5, 37.4]
	2 repeats / systematic error	27.5 [24.1, 30.9]	26.4 [22.4, 30.1]
	4 repeats / independent error	34.4 [31.4, 37.5]	32.8 [29.4, 36.2]
	4 repeats / systematic error	25.1 [22.1, 28.0]	25.0 [21.5, 28.5]
	8 repeats / independent error	29.3 [26.2, 32.6]	27.2 [23.7, 30.9]
	8 repeats / systematic error	20.1 [17.4, 22.5]	20.8 [17.7, 24.1]

H.2 Additional Cross-Task Results

E33–E38 provide additional cross-task tests. Releasing verified local knowledge from an unsuccessful donor improves Cybench success from 28.6% when withheld to 35.3%; combining two compatible experience sets outperforms either alone, and immediate publication outperforms delayed or absent publication. Increasing memory capacity helps only when paired with relevance filtering, while global action blacklists underperform conditional negative evidence. Transfer follows shared dynamics more strongly than surface naming: same dynamics remain useful even when names differ, whereas same names with different dynamics do not. Table 13 gives all cells.

Table 13: Stratified mechanism results with two fixed model families per environment. Cells report terminal success rates and intervals (%).

ID	Controlled condition	Cybench	CVE-Bench
E33	Verified local knowledge from failed donor	35.3 [31.5, 39.5]	30.3 [27.2, 33.7]

Continued

ID	Controlled condition	Cybench	CVE-Bench
	Failed donor knowledge withheld	28.6 [24.9, 32.7]	23.6 [20.3, 27.0]
	Unrelated failed donor	28.5 [24.9, 32.3]	22.8 [19.8, 26.0]
	Continue target at equal cost	32.9 [28.9, 37.1]	28.8 [25.4, 32.3]
	Compatible successful donor	37.2 [32.9, 41.7]	32.8 [29.3, 36.6]
E34	Neither source	24.2 [21.2, 27.6]	23.2 [20.1, 26.6]
	Source A	28.1 [25.0, 31.6]	27.9 [24.5, 31.8]
	Source B	28.9 [25.3, 32.6]	28.1 [24.7, 31.6]
	A + B, equal read budget	38.7 [35.0, 42.2]	40.4 [36.6, 44.3]
E35	Immediate full success publication	38.8 [34.9, 42.6]	36.5 [33.3, 39.6]
	Only local effects published	36.2 [32.6, 39.7]	34.8 [31.1, 38.3]
	Delayed full publication	33.2 [29.7, 36.5]	30.5 [27.2, 33.9]
	No publication	31.2 [28.0, 34.6]	27.3 [24.1, 30.6]
E36	8 entries / random	31.6 [28.0, 35.4]	27.2 [24.1, 30.3]
	8 entries / relevant	32.4 [28.8, 35.8]	28.9 [25.9, 31.9]
	32 entries / random	34.5 [30.3, 38.5]	31.6 [28.1, 35.4]
	32 entries / relevant	37.2 [33.2, 41.5]	35.4 [31.4, 39.5]
	128 entries / random	31.0 [27.6, 34.5]	28.1 [24.9, 31.4]
	128 entries / relevant	38.0 [34.1, 41.9]	34.9 [30.9, 38.8]
E37	No negative evidence	33.6 [30.2, 36.7]	29.0 [25.7, 32.0]
	Conditional negative evidence	37.1 [33.7, 40.4]	34.1 [30.3, 37.8]
	Conditional retrieval / reflection	37.4 [34.0, 40.6]	32.6 [29.2, 35.9]
	Global action blacklist	28.5 [25.8, 31.3]	24.2 [20.8, 27.5]
E38	name same, dynamics same	39.2 [36.1, 42.1]	34.4 [30.5, 38.0]
	name same, dynamics different	29.4 [26.4, 32.0]	25.4 [22.1, 28.8]
	name different, dynamics same	37.8 [34.6, 40.9]	32.2 [28.8, 35.5]
	name different, dynamics different	28.3 [25.3, 31.1]	24.2 [21.1, 27.5]

I Information and Safety Boundary

Strict all-failure runs exclude successful demonstrations, hidden evaluator state, reference solutions, and target-success retrieval; terminal completion is determined only by the fixed task verifier. Authorized isolated environments, task-level provenance, and source/checkpoint lineage are retained as run metadata. Any detected information-boundary violation invalidates the affected strict-cold-start run rather than being repaired by clearing visible memory. These conditions define admissible evidence; they are not reported as separate experimental results.