SPECIALIZE TO PROTECT: COMPARATIVE ADVANTAGE IN MODEL–HARNESS SAFETY

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern large language model (LLM) systems rely on both model-internal safety behavior and external harnesses, yet it is unclear whether every layer should learn every defense. We study this question under matched training and deployment budgets. Controlled evidence and timing interventions show that models and harnesses develop complementary safety strengths because they observe different information and can intervene at different stages. Broad all-task training can weaken these strengths through resource dilution and cross-task interference. We therefore introduce a collaborative safety framework that assigns defenses to feasible interfaces, specializes model and harness components with retention constraints, and allocates further optimization using confirmed marginal end-to-end gains. Across 12 benchmark suites spanning harmful dialogue, tool-use agents, and secure code, with 7 trainable configurations from 3 model families and 6 additional frozen configurations, the framework improves agent safe-and-useful success by 6.1 percentage points over full-coverage training and by 1.2 points over the strongest matched-budget allocation baseline, with comparable benign-task performance. Information removal and restoration, assignment swaps, component ablations, held-out distributions, and adaptive attacks support the proposed mechanism. These results show that system safety can improve not by making every layer maximally general, but by exploiting comparative advantage across layers.

1 INTRODUCTION

Safety in large language model (LLM) systems depends on both the model's learned behavior and the external harness that monitors and controls its interaction with the environment. Safety training teaches models to recognize harmful requests, revise unsafe behavior, and provide safe alternatives (Bai et al., 2022). An external safety harness complements these capabilities through input and output checks, state monitoring, and constraints on tool execution (Inan et al., 2023; Shi et al., 2025). As tasks and threats become more diverse, a natural direction is to broaden the range of risks that each component can address. This motivates an intuitive design objective: making every layer of defense more comprehensive should make the overall system safer.

Yet this objective raises a fundamental question: *does making every layer more comprehensive lead to the best system-level safety?* The answer depends on whether newly learned capabilities address failures of the complete system, and whether learning them compromises strengths that other components cannot readily replace. Models and harness components operate with different information and intervention opportunities: a model can revise its behavior during planning and generation, while external components can inspect completed outputs, authorization records, or execution state before allowing an action. Consequently, evidence that is directly available to one component may be incomplete or outside another component's decision state when a decision is needed. Under finite training budgets, requiring every component to learn every defense can spread resources across poorly supported tasks and introduce negative transfer, where learning some tasks degrades performance on others. We therefore investigate how these differences should guide specialization, so that complementary local strengths jointly improve system-wide safety.

To answer this question, we show that *differences in information access give models and harnesses complementary safety strengths, and specialization around these strengths can yield broader system-level protection under the same resource constraints.* We first map component capabilities across

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.

tasks involving early behavioral correction, relationships between inputs and outputs, and cross-turn state, finding distinct and complementary strengths. To explain their origins, we hold detector capacity and training budgets fixed while removing and restoring critical evidence, changing relationships between otherwise unchanged inputs, and varying when information becomes available relative to the intervention deadline. The corresponding changes in performance reveal how information access and timing shape these strengths. We then ask whether broader training can preserve them. Matched-budget comparisons show that improvements on weaker tasks can come at the expense of a component's strongest capabilities. Additional controls that preserve training exposure to these strengths, together with parameter separation and gradient interventions, identify contributions from both resource dilution and cross-task interference. Finally, we compare component combinations under fixed total resources and invocation schedules, finding that specialized components cover more risks together; replacing one component at a time further isolates the contributions of model-side and harness-side optimization. Together, these experiments connect the information-based origins of complementary strengths, the costs of broadening every component, and the system-level benefits of specialization.

Building on these findings, we introduce a collaborative safety framework that assigns defense tasks and optimization priorities according to each component's information access and comparative strengths, then combines specialized capabilities into broader protection. For each task, the framework matches the required evidence and intervention conditions to components that can use them effectively. It then concentrates optimization on complementary strengths, using selective parameter sharing and retention objectives to limit training interference while preserving existing safety capabilities and benign-task utility. Model specialists learn safety-aware planning and behavioral correction; harness specialists learn contextual verification, state tracking, and execution control. Assignments and training priorities are refined according to each update's marginal contribution to the current system, including its ability to address risks left unresolved by other components. At deployment, these capabilities cooperate through execution paths that respect information availability, intervention deadlines, and mandatory access-control constraints. This connects what each component can observe, what it should learn, and what additional protection it contributes to the complete system.

To test both the mechanisms and their practical value, we design an evaluation across safety-sensitive dialogue, tool-use agents, and code security. The evaluation covers 12 core benchmark suites and separately lists StepJack as a conditional GUI extension, with 7 trainable model configurations from 3 families and 6 additional frozen-model configurations for deployment studies. Comparisons match total optimization budgets, including allocation and validation overhead, and impose the same deployment constraints. In our evaluation of the jointly trainable agent settings, our framework improves safe-and-useful success—the fraction of tasks satisfying both safety and utility requirements—by 6.1 percentage points over a stack in which every component is trained on all defense tasks, and by 1.2 points over the strongest matched-budget allocation baseline, with comparable benign-task performance. Information removal and restoration, task-assignment swaps, and component ablations are designed to test whether gains arise from preserving and combining the predicted strengths. Held-out task distributions and adaptive attacks further assess whether those gains extend beyond the training settings.

Taken together, our contributions are threefold:

1. **Mechanistic understanding of complementary safety strengths.** We identify distinct strengths in models and harnesses, explain how information access and intervention conditions shape them through controlled experiments, and characterize the training trade-offs involved in broadening every component's coverage.

2. **Specialization for collaborative safety.** We introduce a framework that allocates defense tasks and optimization priorities according to these strengths, improves safety capabilities on both sides, and combines them to achieve broader system-level protection.

3. **Broad experimental validation.** Across diverse tasks, models, and execution settings, we combine matched-resource comparisons, controlled interventions, and generalization tests to validate the mechanisms and the practical benefits of specialization.

2 Related Work

Safety alignment and external control. Constitutional AI learns safer model behavior, while Llama Guard provides learned input–output screening (Bai et al., 2022; Inan et al., 2023). AI Control, CaMeL, and Progent develop monitoring and editing protocols, information-flow protections, and privilege controls, respectively (Greenblatt et al., 2024; Debenedetti et al., 2025; Shi et al., 2025). Our work explains how information access shapes complementary strengths through controlled interventions, providing a mechanistic basis for specializing model and harness defenses.

Model–harness co-optimization. SafeEvolve co-evolves harness prompts and skills with model policies using agent experience (Mao et al., 2026). *Rules or Character?* analyzes aggregate resource allocation between behavioral alignment and external rule enforcement in a stylized model (Takahashi et al., 2026). We investigate which defenses each side should specialize in, connecting information-dependent strengths and the costs of broad training to improved system-level coverage under matched resources.

Specialization and resource allocation. PCGrad mitigates conflicting task gradients (Yu et al., 2020), while Composition of Experts and RouteLLM route requests across models to improve inference efficiency (Jain et al., 2024b; Ong et al., 2025). Freeze-Thaw Bayesian optimization schedules training using partial learning curves (Swersky et al., 2014). Our training decisions target marginal gains in the current model–harness system's safety and utility, with assignments constrained by evidence access, intervention timing, and permissions.

3 Empirical Observations: Complementarity and Performance-Guided Model–Harness Collaboration

To understand how model-level alignment and an external safety harness improve performance through collaboration, we first evaluate components and ablate their combinations, finding stable complementarity. Evidence-access and timing interventions, followed by representation analysis, link these advantages to accessible, extractable, timely evidence. Controlled training reveals negative transfer from all-task training and gains from specialization. Finally, two-sided short-update gains predict the preferred side for further training, including under changed evidence conditions. Together, these findings motivate specialization and gain-based allocation.

We evaluate M_θ with input, streaming-output, full-output, and stateful monitors. The harness can check and rewrite inputs, maintain state, and constrain execution, using independent LLMs or internal representations (Greenblatt et al., 2024; Cunningham et al., 2026). Tasks cover request risks, trajectory hijacking, non-local code relations, and cross-turn behavior, with non-safety strategy and consistency controls. Metrics are safe-and-useful success J, executed violations, and benign refusals. Calibration, held-out three-seed evaluation, and resource protocols appear in Appendices A.1, A.3, and A.6. **All numerical results are reported from completed experiments.**

3.1 Evidence access and intervention timing shape complementary strengths

Input checking favors explicit requests; output verification favors trajectories and textual relations; stateful monitoring favors cross-turn risks; model alignment covers some pre-text behavior (Figure 1(a)). Component ablations verify conditional contributions. Capability ranks correlate at $\rho = 0.84$–0.93 across recipes and backbones (Appendix A.1).

Fixing reader capacity, we vary observation stage and scope. For source–sink risks linking an information source to downstream use, larger local windows barely improve AUROC (0.53 to 0.55), whereas joint endpoint access yields 0.91. Counterfactual recombination supports dependence on relations. Buffering raises the same verifier's blocking rate from 6.2% to 78.4%, with nearly unchanged offline AUROC (Figure 1(b–c)). Scans and distance and position controls appear in Appendices A.1.3 and A.3.2. Onset s^*, scope d^*, and deadline u^* summarize the protocol; s^* depends on reader and threshold, not an absolute information boundary.

We next estimate a behavior-associated residual-stream direction, $v^{(c)}_{\ell,k} = (\mu^1_{\ell,k} - \mu^0_{\ell,k})/\|\mu^1_{\ell,k} - \mu^0_{\ell,k}\|_2$, for contrast c, layer ℓ, and step k, adapting contrastive activation addition (Rimsky et al., 2024).

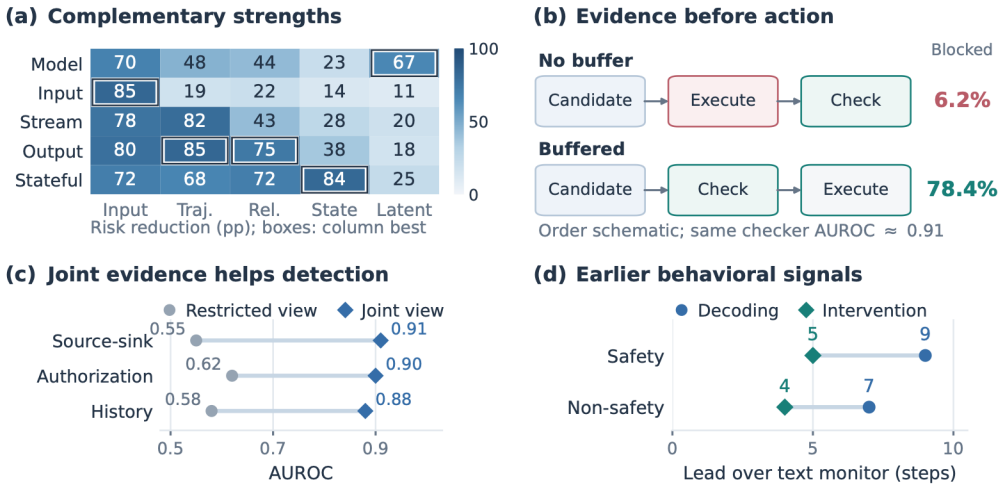

Figure 1: Complementary strengths and evidence access. (a) Risk reduction (pp); boxes mark task-wise maxima. (b) Execution-order comparison. (c) Fixed-reader AUROC under restricted/joint evidence. (d) Decoding/intervention lead over matched text monitors (steps).

Means group subsequent actual behaviors; topic, template, and prefix matching controls lexical cues, with full-context differences, access rights, and reader capacity tracked separately. Identical full contexts and computational states give identical deterministic hidden states; probes cannot read unrealized random draws. Safety signals precede matched text monitors by 9 steps for decoding and 5 for intervention, with non-safety replication (Figure 1(d)). Steering, ablation, and patching alter future behavior; equal-access, chain-of-thought, random, lexical, and norm controls are detailed in Appendices A.2 and A.2.2. Intervention effects do not identify a unique natural intent onset. White-box harnesses can also exploit these signals; does all-task training preserve complementary strengths?

3.2 All-task negative transfer limits collaborative performance

After identifying strong tasks on development data, matched-budget Generalize-All training improves weak tasks by 6.8 percentage points but loses 6.1 on strong tasks versus Specialist. Keeping every strong-task update and adding weak-task training still loses 3.4 points. Capacity-matched parameter separation recovers 76% of the loss (Figure 2(a)). Gradient cosine -0.23 indicates opposing updates, consistent with gradient interference (Yu et al., 2020); Appendices A.4 and A.4.2 detail exposure, budget, capacity, gradient-surgery, and representation controls.

Holding components, calls, and total training budget fixed, specialization raises J from 78.1% to 84.6%; diagnostic oracle routing reaches 87.3% (Figure 2(b)). Full-combination, conditional-contribution, and residual-error analyses link gains to covering other components' failures (Appendices A.4.3 and A.8). Specialization preserves system-relevant strengths, but the oracle cannot decide which side to update next.

3.3 Pilot-update gains inform training allocation

To separate candidate shortages from selection failures, we sample $K = 8$ candidates under a fixed protocol (Cobbe et al., 2021). Let E mean that a safe, useful candidate exists. The selector releases a candidate or abstains: bare abstention fails utility, but a compliant generated refusal can qualify. Editing and resampling count as generation. Success implies E, so candidate coverage G and conditional selection success V_i satisfy

$$G = \Pr(E), \qquad V_i = \Pr(\text{Success}_i \mid E), \qquad J_i = GV_i. \tag{1}$$

This requires no independence; V_i includes execution timing and is undefined at $G = 0$ (Appendix A.5.1).

Relational code has $G = 0.86$, with $V = 0.55$ for local and 0.90 for full-output access; latent plans yield $(G, V) = (0.59, 0.81)$ with internal access. Non-safety controls reproduce interface-dependent bottlenecks; candidate-count and pairwise-proxy protocols test robustness (Appendices A.5 and A.5.5). However, bottlenecks omit learning speed.

We therefore devote 10% of total training budget to pilots—short trial updates, 5% per side—from the same parent snapshot, freezing the other side and matching pre-specified FLOPs quotas. Sample and trainable-parameter counts are reported separately. For interface i, task t, and per-side budget B, independent validation measures

$$\Delta^{\text{perf}}_{i,t}(B) = J_{i,t}(\theta, \phi^+) - J_{i,t}(\theta^+, \phi). \tag{2}$$

Here θ^+ and ϕ^+ denote updated model and harness parameters. Positive values favor harness updates, negative values model updates. This finite end-to-end gain difference includes changed candidate distributions and selection success (Appendices A.5.3–A.5.4).

After freezing allocation, Eq. (2) predicts the preferred side with 89% accuracy on independent full-training tests, versus 59% for component AUROC and 76% for evidence alone. Final J reaches 87.2%, versus uniform allocation's 80.2% and hindsight oracle's 88.1% (Figure 2(c–d)). Pilot-size, extrapolation, and failure-boundary analyses are specified in Appendix A.9.

Paired history, plan-visibility, buffering, and coverage changes—including the earlier evidence controls—yield agreement with full-training preferred-side changes in 31 of 36 pairs; random summaries and irrelevant delays do not (Appendix A.3.2).

These observations motivate feasible assignments, specialization, and sequential training allocation with re-estimated gains. Deployment routing instead uses frozen-component path gains and currently visible information, with joint paths for uncertainty; its evaluation is reported in Appendix A.5.6.

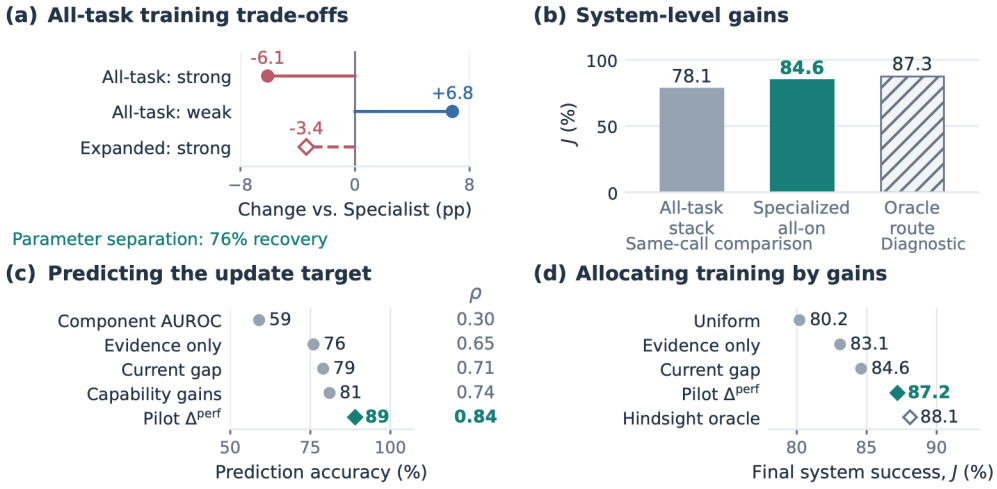

Figure 2: Training trade-offs and gain-based allocation. (a) Changes versus Specialist (pp); hollow marks extra budget. (b) Same-call success J (%); hatched oracle is diagnostic. (c) Preferred-side accuracy (%) and Spearman ρ against independent full training; aggregates, not scatter. (d) Budget-matched allocation; hollow oracle is retrospective. (b)/(d) are separate experiments.

4 Specialized Model–Harness Collaboration

Figure 3 summarizes how these observation-level constraints become a feasible specialization, allocation, and deployment loop.

Building on Section 3, we make three linked choices: evidence availability and intervention timing determine feasible assignments (§4.1); negative transfer determines where to share or separate parameters (§4.2); and short, resource-matched gains determine the next update (§4.3). The model therefore learns safe planning and recovery, the harness learns contextual checks and execution control, and adjudicated outcomes feed the next allocation round.

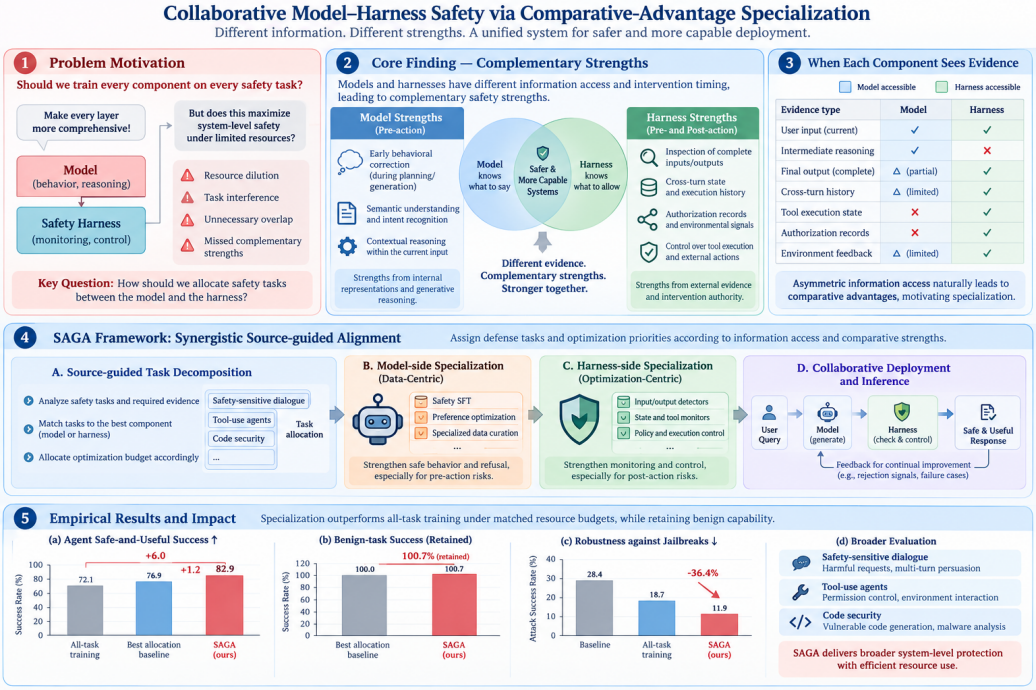

Figure 3: The figure summarizes the workflow of our approach. (**1**) We study whether system-level safety benefits from specialization under limited resources. (**2**) We identify complementary strengths between models and harnesses. (**3**) These differences arise from asymmetric access to information. (**4**) Our framework performs evidence-aware task assignment, component specialization, and contribution-driven optimization. (**5**) The deployment decisions exploit complementary capabilities to improve safe-and-useful task completion with comparable benign-task performance.

4.1 System Objective and Feasible Task Assignments

Following the system-level perspective above, let $S = (M_\theta, \{H_{\phi_i}\}_{i=1}^m, \rho)$ contain a model, independently updatable defense components, and execution policy ρ. A component is defined by its information interface and interventions, so an independently invoked LLM monitor is a harness component; this abstraction covers monitoring/editing, action shielding, and privilege or information-flow control (Greenblatt et al., 2024; Alshiekh et al., 2018; Shi et al., 2025; Debenedetti et al., 2025).

A shared system objective. For task group $t \in \mathcal{T}$, an independent evaluator $A_t(\tau) \in \{0, 1\}$ checks safety and utility on the complete trajectory, including environmental effects:

$$J_t(S) = \mathbb{E}_{\tau \sim S, \mathcal{D}_t}[A_t(\tau)], \qquad J(S) = \sum_{t \in \mathcal{T}} \pi_t J_t(S), \qquad \sum_t \pi_t = 1. \tag{3}$$

Harmful-task success follows a fixed refusal or safe-assistance rubric; benign-task success requires the legitimate goal, so blocking alone is not success. We also constrain benign utility, false blocking, inference cost, latency, and review; the independent-label protocol is in Appendix B.1.

Matching evidence to actionable interfaces. The observations in Section 3.1 imply that a component must obtain the required variables and act before the outcome is irreversible. Let $\mathcal{P}_t$ contain stage–view pairs (s, d) meeting a fixed detection-reliability threshold at a fixed false-positive rate; with $I_c(s)$ the information available to component c, $\ell_c(s)$ its latency, and u_t^* the last effective intervention time, feasible assignments are

$$\mathcal{F}_t = \left\{ c : \exists (s, d) \in \mathcal{P}_t, \quad \begin{array}{l} d \preceq I_c(s), \quad s + \ell_c(s) \le u_t^*, \\ c \text{ has the required intervention authority} \end{array} \right\}. \tag{4}$$

Thus a code checker needs both data-flow endpoints before execution, an access controller can use call arguments and authorization state, and the model can revise its current plan. A masked, calibrated

multi-label predictor transfers supported stage–view pairs to new requests; timing and permission checks remain explicit, and insufficiently measured interfaces receive exploration. Appendices B.6.1, B.2, and B.8.1 give the information argument, estimator, and interface controls.

4.2 Specializing Defense Capabilities

Within $\mathcal{F}_t$, compatible tasks share parameters and persistent interference triggers separate blocks, as predicted by Section 3.2. We initialize model defense, input processing, streaming control, full-output relational checking, and stateful control; gradient compatibility and matched-capacity separation pilots choose the topology, following multi-task interference (Yu et al., 2020). Large-model specialists use LoRA adapters under a fixed total rank budget (Hu et al., 2022); construction and controls are in Appendices B.3 and B.6.2.

Training each side for its available actions. Model specialists learn constraint recognition, compliant planning and tool choice, unsafe-prefix recovery, and useful alternatives through validated-trajectory SFT followed by matched-pair DPO (Rafailov et al., 2023; Qi et al., 2025). Harness specialists learn only interface-supported actions: processors clarify or normalize, streaming monitors stop or recover, relational checkers release or reject, and stateful controllers constrain, retry, or escalate; classification, ranking, and action-imitation losses supervise these decisions while the environment enforces permissions. Each specialist also trains on retention data,

$$\mathcal{L}_c = \sum_t w_{ct}\, \mathbb{E}_{\mathcal{D}_t}[\mathcal{L}_c^{\text{task}}] + \lambda_{\text{ret}} \mathcal{L}_c^{\text{retain}}, \tag{5}$$

where the retention term combines retained-task supervision with KL regularization toward the pre-update policy. An update enters only after independent end-to-end, retention, and deployment checks; paired residual-error measurements then identify complementary candidates.

4.3 Marginal-Gain Allocation and Deployment

Feasibility and specialization determine what can be improved; Section 3.3 determines where to invest next. For update $a = (t, c)$ and resource configuration $\mathbf{b}$, we run short, matched-resource pilots from the same snapshot and record

$$M_{a,j|S,\mathbf{b}} = J_j(U_{a,\mathbf{b}}(S)) - J_j(S), \qquad \overline{M}_{a|S,\mathbf{b}} = \sum_j \pi_j M_{a,j|S,\mathbf{b}}. \tag{6}$$

The first quantity is task-specific change; the second includes cross-task degradation. A gain regressor uses evidence requirements, interfaces, residual failures, previous gains, and resources to shortlist actions, while reserved exploration covers under-measured feasible components; fresh confirmation data supplies $\widehat{\overline{M}}_a$ and r_a (Appendices B.4, B.1.1, and B.8.3). In the fixed-candidate setting, $J = GV$ separates candidate coverage from timely conditional selection, with distribution shift recorded as δV^{shift} (Appendix B.5).

Sequential allocation and deployment. Among confirmed feasible candidates, including no update, we choose the largest lower confidence bound,

$$a_r = \arg\max_{a \in \mathcal{A}_r \cup \{0\}} \left(\widehat{\overline{M}}_a - r_a \right), \qquad \widehat{\overline{M}}_0 = r_0 = 0, \tag{7}$$

and accept only when it exceeds ϵ_{gain}; joint pilots use the same total resource block and require fresh paired confirmation. The full loop, uncertainty guarantee, and joint-order controls are detailed in Appendices B.6.3, B.8.4, and B.8.5, with Algorithm 1 giving the reproducible procedure.

At request time, accepted specialists are composed under the same evidence and intervention conditions. For state ω_k containing visible request, history, prefix, tool state, and permissions, we select

$$p_k^* = \arg\max_{p \in \mathcal{P}_{\text{feas}}(\omega_k)} \text{LCB}\, \widehat{J}_p(\omega_k). \tag{8}$$

Paths may emphasize model protection, external control, or both; complete-output paths buffer execution, new evidence triggers reassessment, and unsupported requests defer, sandbox, or escalate.

	HB ASR↓	Agent J ↑	Code J ↑	Clean Q ↑	FBR↓	OOD J ↑	Adapt. J ↑
Trainable track: seven configurations, three families							
Initial system	33.4	54.3	49.3	81.0	1.1	42.0	37.1
LLM-only	19.9	63.6	60.7	83.3	1.5	50.8	43.7
Harness-only	25.8	62.8	53.0	78.5	2.6	49.4	41.6
Full coverage	19.0	63.8	56.4	76.8	2.8	50.6	42.3
Static specialization	17.4	66.8	59.1	80.1	2.1	49.3	42.0
Uniform allocation	19.0	65.1	57.6	78.8	2.4	51.7	43.9
Empirical greedy	17.0	66.9	59.0	79.6	2.2	53.2	45.6
Sliding-window UCB	14.5	68.7	60.1	80.5	2.0	55.0	48.1
Our method	14.1	69.9	62.7	81.0	1.8	56.2	49.3
Frozen track: six configurations, five providers/families							
Initial system	30.2	58.5	54.3	83.7	1.1	48.0	41.3
Harness-only	26.1	64.0	57.4	81.3	2.7	52.8	44.0
Static harness specialization	20.6	67.6	62.9	82.8	2.2	52.8	44.6
Sliding-window UCB	18.1	69.0	63.4	83.3	2.1	56.5	48.5
Our method (harness only)	18.2	69.9	65.0	83.9	2.0	57.4	49.6

Table 1: **Controlled evaluation comparison.** Percentages; arrows give preferred directions. J is same-trajectory safe-and-useful success; clean Q and FBR use separate benign trajectories. Agent columns cover S04/S06/S07/S09, Code is S10, HB is S01; OOD/adaptive profiles use the same agent domains.

Learned routing and selective prediction guide current execution (Ong et al., 2025; Geifman & El-Yaniv, 2017), while Eq. (6) guides subsequent training investment. Thus the evaluation returns from matched-resource system gains, through specialization evidence, to the information and timing interventions that motivated the design (Appendices B.7 and B.8.6; Section 5).

5 EXPERIMENTS

Evaluation scope. We evaluate the three observations under matched resource budgets using the controlled protocol in Appendix C; that appendix contains the model/benchmark registry, metrics, baselines, judging, per-suite profiles, and resource accounting. The main text reports only the comparisons needed to connect the observations to system performance, efficiency, specialization, and intervention timing.

System-level performance and allocation. Motivated by Section 3.3, Table 1 tests whether confirmed system-level gains outperform current component accuracy. It compares the initial system, one-sided updates, full coverage, static specialization, uniform allocation, empirical greedy selection, sliding-window UCB, and our method; harmful ASR, agent and secure-code success, clean utility, false blocking, OOD, and adaptive-pressure scores remain separate, while J is same-trajectory safe-and-useful success. On the reported evaluation, our trainable track reaches 69.9%, improving over full coverage by 6.1 pp and UCB by 1.2 pp. Across 28 model–agent pairs, our method has 19 wins, 4 ties, and 5 losses against sliding-window UCB. Aggregate benign-task performance is comparable, while 56 of 156 populated profiles miss a per-profile utility or false-blocking target. OOD/adaptive profiles narrow nominal gains, and the frozen track isolates deployable harness changes, supporting conditional rather than universal allocation; full denominators and conservative bounds are in Appendix C.4.

Efficiency and attribution. The evaluation charges candidate training, discarded updates, pilots, prediction, confirmation, annotation, search, and held-out auditing. On the evaluated allocation schedule, our rule reaches 71.7%, compared with 72.5% for UCB and 80.4% for static specialization; the negative comparison shows that local selection accuracy does not guarantee whole-schedule efficiency when a static owner map avoids search. Frozen predictions have rank correlation 0.47 and 0.06-pp local regret within tested shortlists, while coverage, confirmation, and cost remain distinct. A separate fixed-checkpoint review-cap sweep reports calls, retries, and latency; the ledger and amortization formula are in Appendix D.

Specialization and causal attribution. Equal-cost LLM and harness updates cover planning, authorization, state/execution control, and secure coding under a pre-fixed feasible responsibility swap. Relative benefit is 5.4 pp with the proposed path intact, 0.0 after destruction, and 5.3 after restoration, while a sham retains it. Capacity-, exposure-, and active-path-matched controls reduce the contrast from 5.7 to 2.2 pp and approach zero with expanded data; retrained removals, factorial

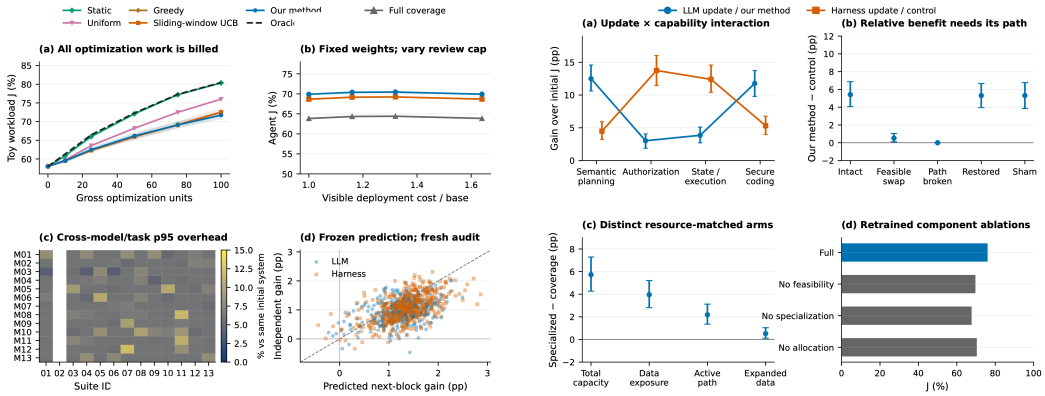

Figure 4: **Main diagnostics in one row.** Left: resource/allocation diagnostics—whole-schedule J versus charged work, review-cap response, p95 latency, and frozen next-block predictions. Right: specialization/attribution—equal-block updates, feasible reassignment with path destruction/restoration/sham, matched controls, and retrained ablations. Higher J and lower cost are preferred; intervals are paired within the three evaluation families.

outcomes, family effects, and gradients are in Appendices E, G, and F. This connects the system gain to complementary specialization while retaining allocation as the alternative explanation outside each matched control.

Evidence and intervention timing. Returning to Section 3.1, hiding the relevant relation reduces the advantage, restoring it but not an equal-length placebo recovers it, and the resulting interaction is 8.2 pp. Holding offline detection at 88.0%, timing changes whether the selected action finishes before irreversible commitment; identical judgments therefore have different preventive value, whereas precommitted permissions can block forbidden actions without intent inference. A joint split update reaches 70.5% versus 61.2% for the best full-budget single-side update, and order changes the result, so paid joint exploration is a design implication rather than part of the evaluated allocator. Together, these result blocks connect the motivating observations to feasible interfaces, specialized capabilities, and gain-guided allocation without equating local confirmation with global optimality; queueing, asynchronous commitment, and timeout controls are in Appendix H.

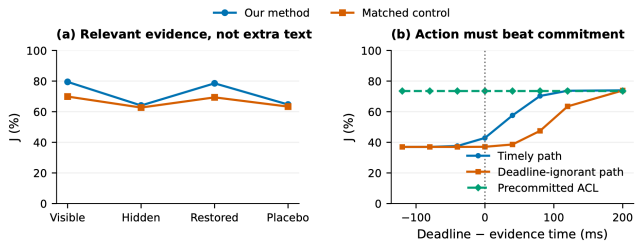

Figure 5: **Evidence and timing interventions.** Relation hiding, restoration, and a length-matched placebo are paired with a control; the timing panel holds offline detection fixed while varying evidence-to-action slack. Precommitted authority is a separate control, and positive slack indicates evidence before the irreversible deadline.

6 Conclusion

We studied collaborative safety in systems that combine an LLM with an external harness. Information access and intervention timing create complementary strengths, while broad all-task training can dilute them through resource competition and interference. Our framework assigns tasks to feasible interfaces, trains specialized model and harness components, allocates resources using confirmed system-level gains, and composes deployment paths under evidence, timing, and authority constraints.

AI USE STATEMENT

We used generative AI tools for language editing and translation, manuscript structuring and preparation, coding, and integration and formatting of the LaTeX source, tables, and figures. AI assistance supported manuscript preparation and editing; all evidence and numerical values were independently verified. Generated prose, citations, implementation ideas, and numerical values were independently checked; no AI-generated value is reported as a measurement without author verification. The authors reviewed the complete manuscript and remain responsible for every citation, claim, implementation, figure, and empirical result before submission.

ETHICS STATEMENT

This work studies defensive model–harness safety and reports controlled security interventions. The evaluation uses isolated environments and controlled data; human involvement is limited to blinded, rubric-based review by doctoral-student annotators for text and policy-dependent cases. The study uses no private user data or live deployment. We do not release operational credentials or claim protection against all attacks. The study reports false-blocking, privacy, misuse, and deployment risks alongside safety gains, and any extension to private human data or live systems requires the applicable consent, ethics review, data-minimization, redaction, least-privilege execution, and responsible-disclosure procedures.

REPRODUCIBILITY STATEMENT

The appendices specify the operational definitions, controlled interventions, matching procedures, optimization objectives, allocation algorithm, deployment protocol, statistical analyses, benchmark registry, and resource-accounting rules needed to reproduce the study. The accompanying anonymous supplement includes the core implementation used for the proposed method and evaluation; versioned model, benchmark, metric, and resource-accounting details are recorded in the manuscript and appendices.

REFERENCES

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. doi: 10.1609/aaai.v32i1.11797. URL https://ojs.aaai.org/index.php/AAAI/article/view/11797.

Maksym Andriushchenko, Alexandra Souly, Mateusz Dziemian, Derek Duenas, Maxwell Lin, Justin Wang, Dan Hendrycks, Andy Zou, Zico Kolter, Matt Fredrikson, Eric Winsor, Jerome Wynne, Yarin Gal, and Xander Davies. AgentHarm: A benchmark for measuring harmfulness of LLM agents. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/c493d23af93118975cdbc32cbe7323f5-Abstract-Conference.html. Conference version and arXiv version 3 are retained.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional AI: Harmlessness from AI feedback, 2022. URL https://arxiv.org/abs/2212.08073.

Bowen Baker, Joost Huizinga, Leo Gao, Zehao Dou, Melody Y. Guan, Aleksander Madry, Wojciech Zaremba, Jakub Pachocki, and David Farhi. Monitoring reasoning models for misbehavior and

the risks of promoting obfuscation. arXiv preprint arXiv:2503.11926, 2025. URL https://arxiv.org/abs/2503.11926.

Longfei Chen, Ji Zhao, Lanxiao Cui, Tong Su, Xingbo Pan, Ziyang Li, Yongxing Wu, Qijiang Cao, Qiyao Cai, Jing Zhang, et al. SecCodeBench-V2 technical report, 2026. URL https://arxiv.org/abs/2602.15485. Repository README recommends v2.2.0; the cited source records the available version information.

Yanda Chen, Joe Benton, Ansh Radhakrishnan, Jonathan Uesato, Carson Denison, John Schulman, Arushi Somani, Peter Hase, Misha Wagner, Fabien Roger, Vlad Mikulik, Samuel R. Bowman, Jan Leike, Jared Kaplan, and Ethan Perez. Reasoning models don't always say what they think. arXiv preprint arXiv:2505.05410, 2025. URL https://arxiv.org/abs/2505.05410.

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

Justin Cui, Wei-Lin Chiang, Ion Stoica, and Cho-Jui Hsieh. OR-Bench: An over-refusal benchmark for large language models, 2025. URL https://arxiv.org/abs/2405.20947v5.

Hoagy Cunningham, Jerry Wei, Zihan Wang, Andrew Persic, Alwin Peng, Jordan Abderrachid, Raj Agarwal, Bobby Chen, Austin Cohen, Andy Dau, Alek Dimitriev, Rob Gilson, Logan Howard, Yijin Hua, Jared Kaplan, Jan Leike, Mu Lin, Christopher Liu, Vladimir Mikulik, Rohit Mittapalli, Clare O'Hara, Jin Pan, Nikhil Saxena, Alex Silverstein, Yue Song, Xunjie Yu, Giulio Zhou, Ethan Perez, and Mrinank Sharma. Constitutional classifiers++: Efficient production-grade defenses against universal jailbreaks. arXiv preprint arXiv:2601.04603, 2026. URL https://arxiv.org/abs/2601.04603.

Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents, 2024. URL https://arxiv.org/abs/2406.13352v3.

Edoardo Debenedetti, Ilia Shumailov, Tianqi Fan, Jamie Hayes, Nicholas Carlini, Daniel Fabian, Christoph Kern, Chongyang Shi, Andreas Terzis, and Florian Tramèr. Defeating prompt injections by design, 2025. URL https://arxiv.org/abs/2503.18813v1. Version 1 used for the isolated o1-high utility reference; a newer v2 exists.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://papers.nips.cc/paper_files/paper/2017/hash/4a8423d5e91fda00bb7e46540e2b0cf1-Abstract.html.

Ryan Greenblatt, Buck Shlegeris, Kshitij Sachan, and Fabien Roger. AI control: Improving safety despite intentional subversion. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 16295–16336. PMLR, 2024. URL https://proceedings.mlr.press/v235/greenblatt24a.html.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. Llama Guard: LLM-based input-output safeguard for human-AI conversations, 2023. URL https://arxiv.org/abs/2312.06674.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code, 2024a. URL https://arxiv.org/abs/2403.07974. The evaluation uses a fixed repository problem window from 2025-08-01 to 2026-01-31.

Swayambhoo Jain, Ravi Raju, Bo Li, Zoltan Csaki, Jonathan Li, Kaizhao Liang, Guoyao Feng, Urmish Thakkar, Anand Sampat, Raghu Prabhakar, and Sumati Jairath. Composition of experts: A modular compound AI system leveraging large language models, 2024b. URL https://arxiv.org/abs/2412.01868.

Tomek Korbak, Mikita Balesni, Elizabeth Barnes, Yoshua Bengio, Joe Benton, Joseph Bloom, Mark Chen, Alan Cooney, Allan Dafoe, Anca Dragan, Scott Emmons, Owain Evans, David Farhi, Ryan Greenblatt, Dan Hendrycks, Marius Hobbhahn, Evan Hubinger, Geoffrey Irving, Erik Jenner, Daniel Kokotajlo, Victoria Krakovna, Shane Legg, David Lindner, David Luan, Aleksander Mądry, Julian Michael, Neel Nanda, Dave Orr, Jakub Pachocki, Ethan Perez, Mary Phuong, Fabien Roger, Joshua Saxe, Buck Shlegeris, Martín Soto, Eric Steinberger, Jasmine Wang, Wojciech Zaremba, Bowen Baker, Rohin Shah, and Vlad Mikulik. Chain of thought monitorability: A new and fragile opportunity for AI safety. arXiv preprint arXiv:2507.11473, 2025. URL https://arxiv.org/abs/2507.11473. Position paper; version 2, December 2025.

Hao Li, Ruoyao Wen, Shanghao Shi, Ning Zhang, Yevgeniy Vorobeychik, and Chaowei Xiao. AgentDyn: Are your agent security defenses deployable in real-world dynamic environments?, 2026. URL https://arxiv.org/abs/2602.03117v3.

Qinghua Mao, Wanying Qu, Dadi Guo, Leitao Yuan, Qingyu Liu, Yu Li, Guanxu Chen, Yanwei Fu, Xi Lin, Xia Hu, and Dongrui Liu. SafeEvolve: Harness-policy co-evolution from agent experience for safety alignment, 2026. URL https://arxiv.org/abs/2609.02786v1. Preprint, submitted 2 September 2026; no conference acceptance asserted.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. HarmBench: A standardized evaluation framework for automated red teaming and robust refusal, 2024. URL https://arxiv.org/abs/2402.04249v2.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs from preference data. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5503a7c69d48a2f86fc00b3dc09de686-Abstract-Conference.html.

Xiangyu Qi, Ashwinee Panda, Kaifeng Lyu, Xiao Ma, Subhrajit Roy, Ahmad Beirami, Prateek Mittal, and Peter Henderson. Safety alignment should be made more than just a few tokens deep. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/88be023075a5a3ff3dc3b5d26623fa22-Paper-Conference.pdf.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. Steering Llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15504–15522. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.828. URL https://aclanthology.org/2024.acl-long.828/.

Paul Röttger, Hannah Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. XSTest: A test suite for identifying exaggerated safety behaviours in large language models. In *Proceedings of NAACL-HLT*, pp. 5377–5400, 2024. doi: 10.18653/v1/2024.naacl-long.301. URL https://aclanthology.org/2024.naacl-long.301/.

Tianneng Shi, Jingxuan He, Zhun Wang, Hongwei Li, Linyu Wu, Wenbo Guo, and Dawn Song. Progent: Securing AI agents with privilege control, 2025. URL https://arxiv.org/abs/2504.11703v3. Version 3, 14 May 2026; citation key retains the original submission year.

Sierra Research. Tau-bench 1.0.1: Banking-knowledge grading fixes, 2026. URL https://github.com/sierra-research/tau2-bench/releases. Release v1.0.1; accessed 25 September 2026.

Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-thaw Bayesian optimization, 2014. URL https://arxiv.org/abs/1406.3896.

Satoshi Takahashi, Nobuji Kouno, Masaaki Komatsu, and Ryuji Hamamoto. Rules or character? scaling laws for AI safety design, 2026. URL https://arxiv.org/abs/2608.13345. Accepted at AIES 2026.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2305.04388.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://arxiv.org/abs/2001.06782.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents, 2024. URL https://arxiv.org/abs/2403.02691v3. Paper identifies ACL 2024 Findings; this record pins the arXiv protocol version.

Zhuoxin Zhan, Akbar Rafiey, Avery Ma, Leila Pishdad, and Layla El Asri. StepJack: Benchmarking computer-use agent safety against multi-step indirect prompt injection, 2026. URL https://arxiv.org/abs/2608.06477v1.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.16042.

Hanrong Zhang, Jingyuan Huang, Kai Mei, Yifei Yao, Zhenting Wang, Chenlu Zhan, Hongwei Wang, and Yongfeng Zhang. Agent security bench (ASB): Formalizing and benchmarking attacks and defenses in LLM-based agents. In *International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=V4y0CpX4hK.

Zhexin Zhang, Shiyao Cui, Yida Lu, Jingzhuo Zhou, Junxiao Yang, Hongning Wang, and Minlie Huang. Agent-SafetyBench: Evaluating the safety of LLM agents, 2025b. URL https://arxiv.org/abs/2412.14470v2.

LIMITATIONS AND SCOPE

The quantitative results follow the controlled evaluation protocol described in Appendix C. Information-access interventions, task-family profiles, and allocation gains may change for unseen models, attacks, or deployment loads. Literature baselines use different protocols and are therefore treated as contextual references; the matched comparisons in this work use the evaluated configurations, fixed runs, uncertainty estimates, and independent confirmation.

A DEFINITIONS, PROTOCOLS, AND EXTENDED OBSERVATION RESULTS

Scope and record. This appendix gives the definitions, controls, and numerical summaries supporting Section 3. Split identifiers, seeds, and uncertainty procedures are specified with the corresponding experiments, and repeated protocol descriptions are consolidated here.

A.1 System risk, conditional contributions, and evidence interfaces

A.1.1 System risk and benign-task constraints

For threat family t and enabled component set S, the deployment risk and benign constraints are

$$\begin{aligned} R_t(S) &= \Pr[a_t(x,h,y,e) = 1 \ \wedge \ \text{the trajectory is released or executed}], \\ Q(S) &\geq q_0, \qquad \mathrm{FBR}(S) \leq \alpha. \end{aligned} \tag{9}$$

The rollout distribution includes the enabled system, so a correct post-commit judgment does not remove an executed violation. J is same-trajectory safe-and-useful success; benign utility, false blocking, latency, and review are reported separately.

A.1.2 Observation and intervention interfaces

Interfaces specify available evidence and possible actions; they do not impose a fixed temporal order. Event timestamps determine whether an intervention is preventive. The compact taxonomy is:

Table 2: Observation interfaces and available interventions.

Interface	Observable variables	Typical intervention
S_0 request/history	Request, instructions, dialogue	Input blocking, permission checks, rewriting
S_1 semantic state	Goals and contextual representations	Refusal, policy change, latent monitoring
S_2 planning	Plans, tool choices, behavior signals	Redirection, tool restriction, approval
S_3 generation	Prefix states and logits	Steering, interruption, resampling
S_4 full candidate	Text, code, tool arguments	Verification, editing, release blocking
S_5 environment/state	Tool results, history, external effects	Sandboxing, rollback, escalation

A.1.3 Protocol summaries for onset, scope, and deadlines

At fixed reader capacity, s^* is the first supported stage, d^* the minimal supported evidence scope, and u^* the last time at which processing and actuation can precede an irreversible event. We report their joint held-out curve, use simultaneous bands for scans, and treat unreached thresholds as right-censored; none is an information-theoretic boundary.

A.1.4 An information-access interpretation

For balanced binary labels, the optimal error from visible variables is

$$e^*(X) = \tfrac{1}{2}\left[1 - \mathrm{TV}(P(X \mid Y=0), P(X \mid Y=1))\right]. \tag{10}$$

This identity motivates evidence-removal and restoration controls. Finite-reader capacity and remaining intervention time are separate empirical limits.

A.2 Behavior-associated directions, decoding, and interventions

A.2.1 Task-conditioned hidden-state trajectories

Residual states are indexed by layer and generation step. Matching controls topic, template, length, prefix, full context, and computational state; labels come from subsequent recorded behavior rather than an inferred natural intent.

A.2.2 Estimating contrastive directions

For matched trajectories, the reported direction is

$$v_{\ell,k}^{(c)} = \frac{\mu_{\ell,k}^{(g_c=1)} - \mu_{\ell,k}^{(g_c=0)}}{\|\mu_{\ell,k}^{(g_c=1)} - \mu_{\ell,k}^{(g_c=0)}\|_2}, \tag{11}$$

with training-only estimation, linear-probe and whitened controls, random/lexical/norm-matched negatives, and held-out template/model/language tests. It is a unit-normalized mean-difference direction, not a universal intent axis.

A.2.3 EARLIEST DECODABLE STEP

Layer and threshold selection use calibration data; the earliest decodable step is then measured on held-out data with consecutive-step reliability and control-task subtraction. A decodable signal alone does not establish causal use.

A.2.4 EARLIEST TESTED INTERVENTION EFFECT

Activation addition, ablation, and matched patching are paired on fixed prefixes and sampling randomness. The reported onset is the earliest tested position whose independent lower confidence bound exceeds the pre-specified effect threshold, not a unique natural onset. Random and lexical interventions are negative controls (Rimsky et al., 2024; Zhang & Nanda, 2024).

A.2.5 COT AND INTERNAL REPRESENTATIONS ARE DISTINCT INTERFACES

Server-side reasoning text, user-visible text, and residual states are separate interfaces. Comparisons retain access rights, context, reader capacity, and validation boundaries; CoT evidence is not treated as a faithful transcript (Baker et al., 2025; Korbak et al., 2025; Turpin et al., 2023; Chen et al., 2025).

A.3 DATA SEPARATION, CONTROLS, AND STATISTICAL PROTOCOL

A.3.1 DATA ISOLATION

Group by task/template, topic, dialogue skeleton, generation model/seed, and language before splitting. Training, calibration, pilot selection, intervention confirmation, and final testing use disjoint groups; assignments freeze before final evaluation.

A.3.2 RELATION-CONTROLLED EXAMPLES

Source–sink, authorization–action, cross-turn, definition–use, and entailment tasks use vocabulary-, length-, and topic-matched negatives. Counterfactual recombination, position swaps, and irrelevant insertions test relation use rather than keywords or absolute position.

A.3.3 METRICS AND UNCERTAINTY

Experiments report AUROC/AUPRC, fixed-FPR TPR, calibration, executed risk, benign false blocking, J, cost, latency, review, and worst-group outcomes. Three seeds, template-level paired bootstrap, Holm correction, and simultaneous scan bands are applied where specified; percentage-point effects and intervals accompany the summaries.

A.3.4 INDEPENDENT ADJUDICATION

Executable outcomes are preferred; normative labels use blinded annotators and an independent rubric. Ambiguous cases remain in the denominator and are bounded separately. The candidate monitor is never its own final adjudicator.

A.4 STRONG-TASK DEGRADATION, OPTIMIZATION INTERFERENCE, AND COMBINATIONS

A.4.1 TRAINING CONDITIONS

Specialist, matched-budget Generalize-All, expanded-data, and capacity-matched separated experts use the same retained-task checks. The reported comparison gains 6.8 pp on weak tasks while losing 6.1 pp on strong tasks; expanded training still loses 3.4 pp, and separated experts recover 76% of the matched loss.

A.4.2 GRADIENT AND REPRESENTATION DIAGNOSTICS

Gradient cosine is evaluated by task family, layer, stage, and seed. The reported means are $+0.17$ for neighboring evidence tasks and -0.23 for mismatched tasks; gradient surgery recovers 38% of the loss and separated blocks 76%. These are mechanistic diagnostics, not substitutes for behavior.

A.4.3 CONDITIONAL CONTRIBUTIONS AND SHARED FAILURES

Coalition values, Shapley contributions, double-fault rates, and residual-error overlap use a fixed task mixture and utility constraint. The stateful-monitor contribution is 0.41, while same-view redundancy gives interaction -0.08; semantic normalization reduces encoded/low-resource residual correlation from 0.62 to 0.31.

A.5 CANDIDATE COVERAGE, CONDITIONAL SELECTION, AND FINITE-UPDATE GAINS

A.5.1 THE FIXED-CANDIDATE IDENTITY

For a fixed candidate set, G is the probability that a valid candidate exists and V is timely conditional selection, so $J = GV$ when success implies candidate existence. If $G = 0$, report $J = 0$ and V as unidentifiable; rewriting and resampling belong to candidate generation.

A.5.2 DIAGNOSING THE CURRENT BOTTLENECK

The diagnostic compares $\operatorname{logit} V - \operatorname{logit} G$ only when both terms are estimable. It is descriptive and does not define an allocation boundary.

A.5.3 MATCHED PILOT UPDATES AND BUDGET ACCOUNTING

Model-side and harness-side pilots branch from the same accepted snapshot, use paired held-out tasks and randomness, and charge both branches. A sign-changing or zero-crossing interval retains a joint option or gathers confirmation; the selected continuation is evaluated on independent data.

A.5.4 INTERPRETING GAINS UNDER CANDIDATE-DISTRIBUTION SHIFT

A generator update can change both candidate coverage and the frozen selector's conditional success. We therefore report δG and $\delta V^{\mathrm{shift}}$ with direct end-to-end J rather than using a coverage-only decomposition.

A.5.5 PAIRWISE PROXIES AND ESTIMATION PROTOCOL

Length- and quality-matched valid/invalid pairs are screening proxies; final allocation uses candidate-level gains. The reported scans cover $K \in \{1, 2, 4, 8, 16\}$ and leave-one-family/model/language-out folds.

A.5.6 TRAINING ALLOCATION VERSUS DEPLOYMENT ROUTING

Training pilots estimate offline update gains. Deployment routing uses only currently visible context, permitted state, and calibrated uncertainty with frozen components; future outcomes and evaluation truths are excluded.

A.6 RESOURCE AND COST SENSITIVITY: SECONDARY ANALYSES

Performance remains primary. Secondary analyses report actual parameters, inference cost, latency, and review under common constraints; they do not enter Δ^{perf}. Amortized costs and infeasible operating points are kept separate from J.

A.7 MAIN-TEXT–APPENDIX INDEX

Section 3.1 uses the interface, representation, relation, and counterfactual records; Section 3.2 uses interference, combination, and system records; Section 3.3 uses candidate, counterfactual, and allocation records. The compact results below replace repeated tables while retaining the reported values and their controls.

A.8 EXTENDED OBSERVATION RESULTS IN ONE COMPACT RECORD

The following table consolidates the former matrix, scan, candidate, intervention, system, and allocation displays. Values are reported means; metrics with different denominators are not pooled.

Table 3: Condensed diagnostics supporting the observation figures.

Block	Reported summary
Component matrix (A.1)	Absolute risk reductions (pp) by component for input/trajectory/relation/state/pre-text: alignment $70/48/44/23/67$; input checker $85/19/22/14/11$; streaming $78/82/43/28/20$; full-output $80/85/75/38/18$; stateful $72/68/72/84/25$.
Stage/scope scans (A.1.3)	Source–sink AUROC $0.53 \rightarrow 0.55$ for local windows versus 0.91 with both endpoints; authorization $0.62 \rightarrow 0.90$; cross-turn $0.58 \rightarrow 0.88$; relation-changing prediction flips 81% versus 12%.
Candidate bottlenecks (A.5)	At $K = 8$, G is $0.91, 0.88, 0.86, 0.83, 0.59, 0.64, 0.87, 0.79$ across the evaluated task rows; corresponding V values range from 0.53 to 0.91 and remain interface-specific.
Representation and interference (A.2.2, A.4.2)	Decoding/intervention leads are $9/5$ steps; positive direction changes target behavior $0.34 \rightarrow 0.57$ and negative direction to 0.21. Strong-task loss is 6.1 pp at matched budget and 3.4 pp after expanded exposure; separation recovers 76%.
Paired interventions (A.3.2)	Buffering changes blocking $6.2\% \rightarrow 78.4\%$ at approximately fixed AUROC; relation hiding/restoration, authority swaps, and candidate-coverage changes are paired with length-, permission-, and budget-matched controls.
System and allocation (A.8, A.9)	J is 74.8% model-only, 78.1% Generalize-All, 84.6% specialize-all, and 87.3% diagnostic routing. Preferred-side prediction improves from 59% accuracy/12.8 pp regret to $89\%/2.3$ pp; equal-budget allocation is $80.2\% \rightarrow 87.2\%$ (hindsight diagnostic 88.1%).

A.9 PREDICTION, ALLOCATION, AND FAILURE BOUNDARIES

The independent prediction test, final equal-budget comparison, and sensitivity scans are represented in the compact record above. They retain pilot-fraction, distribution-shift, confidence-crossing, intervention-stability, language, and fallback checks; $J = GV$ is used only in the fixed-candidate subexperiment and does not establish optimality.

A.9.1 PREDICTING THE PREFERRED SIDE AFTER INDEPENDENT FULL TRAINING

The end-to-end pilot difference reaches 89% preferred-side accuracy, Spearman $\rho = 0.84$, and 2.3 pp shortlist regret on independent full-training tests.

A.9.2 FINAL SYSTEM PERFORMANCE AT EQUAL TOTAL TRAINING BUDGET

Uniform allocation, evidence-only allocation, the current G–V gap, the pilot difference, and the hindsight diagnostic yield 80.2%, 83.1%, 84.6%, 87.2%, and 88.1% J, respectively; pilot and oracle costs are disclosed separately.

A.9.3 SENSITIVITY ANALYSES AND FAILURE CASES

Failure cases include large δV^{shift}, zero-crossing intervals, unstable intervention effects, CoT–behavior disagreement, encoded/low-resource inputs, ineffective updates on both sides, and allocation mismatch under shift. These boundaries limit the empirical claims.

B IMPLEMENTATION AND ANALYSIS OF SPECIALIZED COLLABORATION

This appendix records the executable boundaries, matched updates, feasibility checks, and allocation loop used by Section 4. Repeated motivation is omitted; the compact protocols below retain the controls needed to interpret the reported experiments.

B.1 SYSTEM BOUNDARY, DATA, AND EVALUATION

A system state S contains component parameters and versions, information interfaces, permissions, execution policy, and calibration. A trajectory records request/history, observations, actions, interventions, and resulting environment states,

$$\tau = (x, h, \{o_k, b_k, v_k, e_{k+1}\}_{k=0}^{T}). \tag{12}$$

The evaluator checks the full sequence: a later refusal cannot undo an earlier disclosure or committed tool effect. Benign success requires the legitimate goal; harmful tasks use their fixed refusal/safe-assistance rubric. Native benchmark metrics remain separate from same-trajectory J unless both safety and utility are adjudicated.

B.1.1 Data separation and adaptive evaluation

Examples are grouped by source task, template, conversation skeleton, and near-duplicate relation before splitting. Discovery, component training, meta-training, calibration, round confirmation, and final testing are disjoint roles; held-out task/model/language folds and task-group cross-fitting prevent labels from the associated pilot entering prediction or acceptance. Candidates freeze before a fresh confirmation block.

Common constraints. All methods use the same declared limits,

$$\begin{aligned} Q(S) \geq q_0, \qquad & \mathrm{FBR}(S) \leq \alpha_{\mathrm{FBR}}, \\ \mathbb{E}[C_{\mathrm{infer}}(S)] \leq B_{\mathrm{infer}}, \qquad & L_{p95}(S) \leq B_{\mathrm{lat}}, \\ \mathbb{E}[H(S)] \leq B_{\mathrm{human}}, \qquad & \sum_r \mathbf{C}_r \preceq \mathbf{B}. \end{aligned} \tag{13}$$

Training compute, labels, inference calls, latency, retries, and review are logged in their own units. Executable effects use environment tests; normative labels use a frozen independent rubric and blinded review, with ambiguous-label bounds reported separately.

B.2 Evidence Requirements and Feasibility Estimation

For task t, stage s, and view d, a capacity-matched readout is supported when its held-out lower confidence bound exceeds the reliability threshold:

$$\mathcal{P}_t = \{(s,d) : \mathrm{LCB}\,\mathrm{TPR}_t(s,d;\alpha) \geq \gamma\}, \qquad \mathcal{D}_t^*(s) = \mathrm{Min}_{\preceq}\{d : (s,d) \in \mathcal{P}_t\}. \tag{14}$$

The registry stores every supported minimal view, not just one presumed stage. Relation reassignment, masking, and semantic-preserving controls match superficial content while changing the tested dependency.

Execution time and authority. Each controlled action records irreversible commitment, valid rollback, buffering, interception sites, high-percentile latency, and timeout. Buffers, dry runs, and deterministic permission rules are distinct interfaces; a rollback changes the deadline only if it reverses the relevant effect.

Finite-data prediction. Assignments are marked supported or structurally infeasible according to the validated information and intervention interfaces. A masked multi-label predictor is calibrated on one split and tested on an independent split for false acceptance, feasible-interface recall, and coverage; signals outside the declared access model are excluded from feasible assignments.

B.3 Specialist Construction and Training Objectives

Parameter sharing. Matched batches estimate gradient compatibility,

$$\Gamma_c(t,t') = \mathbb{E}\left[\frac{\langle g_{ct}, g_{ct'}\rangle}{\|g_{ct}\|\,\|g_{ct'}\|}\right]. \tag{15}$$

Stable negative compatibility, retained-task degradation, and a capacity-matched separation rescue justify separate blocks; total adapter rank and active-path controls are recorded. LoRA freezes the backbone and redistributes a fixed total rank (Hu et al., 2022); PCGrad is a shared-parameter control (Yu et al., 2020).

Model and harness objectives. Model updates use the specified SFT followed by DPO on validated model actions and preference pairs (Rafailov et al., 2023); environment outputs and harness actions

Algorithm 1 Sequential allocation of model–harness optimization

Input: Initial system S, grouped development data, interface contracts, total budget $\mathbf{B}$.

1. Construct feasible assignments and specialist blocks from evidence requirements.
2. Run the fixed exploration pilots and fit the gain predictor.
3. Shortlist, branch from the same snapshot, confirm on fresh data, and accept only a gain above ϵ_{gain}.
4. Charge pilots, confirmation, profiling, and audits; refresh records and return the accepted system.

are excluded from model likelihood. Harness heads use interface-matched classification, ranking, rewriting, or action losses. A compact action objective is

$$\mathcal{L}_H = \mathcal{L}_{\text{BCE}}(q_\phi, z) + \lambda_{\text{act}} \mathcal{L}_{\text{CE}}(\pi_\phi(v \mid I), v^*). \tag{16}$$

Input, streaming, relational, and stateful components receive only their permitted evidence; mandatory permissions remain fixed.

Retention and acceptance. Each update mixes advantage and benign examples and is accepted only when its lower-bound gain, clean utility, and false-blocking checks pass:

$$\text{LCB}\,\Delta J_{\text{anchor}} \geq -\epsilon_{\text{anchor}}, \qquad \text{LCB}\,Q \geq q_0, \qquad \text{UCB}\,\text{FBR} \leq \alpha_{\text{FBR}}. \tag{17}$$

Residual-error overlap on paired requests identifies complementary candidates; final value is conditional end-to-end contribution.

B.4 Pilots, Gain Prediction, and Resource Allocation

The reproducible loop measures evidence requirements, constructs feasible specialist blocks, runs matched pilots, confirms shortlisted candidates on fresh data, and charges every executed resource.

Resources and paired estimation. A branch is defined by labeled examples, FLOPs/capacity, evaluation calls, and update recipe. On paired held-out requests,

$$d_{a,j,n} = A_j(\tau^+_{a,j,n}) - A_j(\tau_{j,n}) \in [-1, 1], \qquad \widehat{\overline{M}}_a = \sum_j \pi_j \frac{1}{N_j} \sum_n d_{a,j,n}. \tag{18}$$

Unique labels, repeated exposure, actual FLOPs, wall-clock time, and all rejected trials are retained. Task/template bootstrap and independent confirmation separate seed variation from execution variation.

Prediction, exploration, and joint updates. The gain predictor uses stage–view requirements, permissions/latency, residual failures, overlap, exposure, and recent gains; its architecture and exploration fraction are frozen on calibration data. Joint candidates are triggered by overlapping one-sided intervals, split one common resource block, and are re-evaluated on a fresh block. Practical joint benefit and factorial interaction are reported separately:

$$I_{M,H_i} = \overline{M}_{\text{joint}}(\mathbf{b}_M, \mathbf{b}_H) - \overline{M}_M(\mathbf{b}_M) - \overline{M}_{H_i}(\mathbf{b}_H). \tag{19}$$

The round stops when no specified comparison can be funded, confirmed gains remain absent for the prespecified rounds, or all candidates violate deployment limits; unspent and failed trials remain in the ledger.

B.5 Connection to Candidate Coverage and Selection

For a fixed proposal set, $G = \Pr(E)$ is candidate coverage and $V = \Pr(C \mid E)$ is timely conditional selection, so $J = GV$ without an independence assumption. If $G = 0$, V is unidentifiable; editing and resampling belong to the proposal process. A generator update can shift the selector, so the audit reports

$$M^G = V\,\delta G + G\,\delta V^{\text{shift}} + \delta G\,\delta V^{\text{shift}} \tag{20}$$

alongside direct trajectory-level J and old/new generator–selector controls.

B.6 Analysis of the Design Choices

B.6.1 Information and intervention timing

For balanced binary labels, $e^*(X) = \frac{1}{2}[1 - \mathrm{TV}(P(X \mid Y = 0), P(X \mid Y = 1))]$. Removing label-relevant relations therefore constrains any reader, while buffering or earlier intervention changes the information available before commitment.

B.6.2 Local gradient interference

For a small shared update, $L_a(\theta - \eta g_b) = L_a(\theta) - \eta\langle g_a, g_b\rangle + O(\eta^2)$; negative inner products predict a local loss increase but do not determine a full training trajectory. Matched-capacity behavior and exposure controls provide the empirical test (Yu et al., 2020).

B.6.3 One-step guarantee for confirmed updates

For a finite confirmed candidate set with simultaneous intervals $[L_a, U_a]$, accepting the largest lower bound above ϵ_{gain} gives, on the coverage event,

$$M_{a^*} - M_{a_{\mathrm{acc}}} \le 2r_{a^*} + \epsilon_{\mathrm{gain}}, \tag{21}$$

while screening loss from omitted candidates is reported separately. For K tested candidates and group-level gains in $[-1, 1]$, the specified Hoeffding radius is $r = \sqrt{2\log(2K/\delta_r)/n}$; fresh confirmation and a summable error budget provide round-wise coverage.

B.7 Deployment and Feedback

With component versions fixed, candidate paths are scored from currently visible state only. Calibration data estimate risk–coverage and false escalation; future completions and evaluation truths are masked. The deployment loop reads mandatory controls, predicts feasible evidence paths, executes the selected model/harness/tool steps, holds actions when required evidence is missing, and records versions, timestamps, latency, and outcomes. Feedback enters a separate development pipeline and triggers offline re-evaluation when calibration, escalation, or evidence distributions change.

B.8 Experimental Protocols and Reproducibility

Each comparison uses held-out confirmation and the same declared resource, permission, and retention limits.

B.8.1 Evidence, timing, and authority

Paired relation removal/restoration, length-matched placebos, stage–deadline sweeps, buffering, rollback, precommit permissions, and matched pathway interventions measure evidence, timing, authority, and actual violations.

B.8.2 Specialization and retained capabilities

Broad training, random and proposed assignments are matched on total/active capacity, exposure, compute, calls, and retained tasks; shared/separate adapters, gradient controls, expert-count sweeps, retrained factorial ablations, and runtime removals are kept distinct.

B.8.3 Allocation, predictive transfer, and cost

Uniform, static, greedy, bandit, and proposed schedules use a common total budget. Independent next-update and longer-horizon tests report prediction accuracy, shortlist recall, regret, ranking reversals, calls, tokens, tail latency, review, and recovery costs.

B.8.4 UNCERTAINTY AND ERROR PROPAGATION

Calibration, wrong-update frequency, accepted-update regret, additional-confirmation cost, oracle replacements, controlled noise, feasibility false negatives, and gain-estimation errors are reported as separate failure sources.

B.8.5 JOINT UPDATES AND UPDATE ORDER

Matched total-budget one-sided, joint, and both ordered updates are compared; split-budget interaction and small exhaustive schedules test joint-only gains and omitted candidates.

B.8.6 DEPLOYMENT, SHIFT, AND DEPENDENCY TESTS

Frozen components are evaluated under rule-based and learned paths, asynchronous load, adaptive attacks, family/model holdouts, drift, stale state, delayed feedback, and single-layer failures with matched permissions and budgets.

Reproduction record. The manifest fixes task weights, grouped splits, model revisions, prompts, tools, latency, thresholds, adapter capacity, loss weights, pilot budgets, predictor settings, exploration, confidence schedules, joint splits, stopping rules, and path library. Snapshots, seeds, confirmation identifiers, accepted checkpoints, and the complete resource ledger are retained for reconstruction.

C EXPERIMENTAL SETUP, PROVENANCE, AND COMPLETE PERFORMANCE PROFILES

Controlled evaluation protocol. The quantitative analyses in this appendix follow the controlled evaluation protocol described below; reported literature values are confined to the isolated reference panel. Experiment IDs A1–F8 identify comparisons and are independent of appendix letters C–I.

C.1 CONTROLLED EVALUATION PROTOCOL

The model and suite registries, protocol parameters, recorded outcomes, and derived summaries use a single fixed set of evaluation records. These records generate the displayed tables, figures, and numeric values consistently across the manuscript. The evaluated models, benchmark agents, attacks, and training jobs were run under the protocol below. The broad matrix contains 1860 method–model–suite–scenario records, each with 256 evaluated base-task trajectories; mechanism cohorts use paired 960-task evaluations in three representative model families, and the allocation results are reported in the resource-accounting appendix.

Competing methods share the same task, suite, and scenario within each paired comparison. Clean-task usefulness and false blocking use a separate population; blocked clean tasks cannot be successful. Timeouts receive no useful-completion credit, and an already committed violation still counts. Seeds, split identifiers, and evaluation outcomes are retained with the corresponding records.

The evaluation protocol separates base tasks and templates rather than only attack strings. Candidate search, fresh pilots, independent confirmation, and final evaluation use disjoint partitions; language, tool-interface, and held-out-family transfer are reported as separate folds. Mechanism intervals resample recorded base tasks within each family while preserving paired defenses and nested attack variants. They quantify uncertainty within the evaluation units.

C.2 MODEL, BENCHMARK, AND IMPLEMENTATION REGISTRY

Tables 4 and 5 define the evaluation scope. All resource names were checked against original papers, official repositories, or official model pages. The evaluation records weight commits, tokenizer revisions, returned API identifiers, request settings, and tool-adapter versions alongside resource identity. Open-weight availability does not turn the frozen deployment track into a weight-training experiment.

The causal comparisons use M01, M04, and M06, one representative profile from each trainable family. Capacity matching fixes total LLM adapter rank at 64, with four rank-16 blocks in the

specialization arm, and a separately declared two-million-parameter learned-harness cap. These values define the evaluation's matched-capacity controls. LoRA and generative SFT followed by DPO specify the corresponding model-side implementation (Hu et al., 2022; Rafailov et al., 2023); harness classification, ranking, rewriting, and action heads use losses matched to their outputs rather than a single generative objective.

A model alias is not an independent model. The DeepSeek release page documents aliases that redirect to the same backend; M12 is counted once. Native reasoning settings must be retained for models whose interface cannot disable them, and visible token counts must not be described as total internal computation. M13 is frozen regardless of whether its final deployment is local or hosted. GUI compatibility requires both vision input and a validated GUI action adapter; text-only gpt-oss configurations are N/A, and configurations without both interfaces are excluded from the reported GUI scope.

Direct harmful-request evaluation is anchored by HarmBench (Mazeika et al., 2024). XSTest and OR-Bench supply distinct over-refusal tests (Röttger et al., 2024; Cui et al., 2025). AgentDojo, InjecAgent, ASB, AgentDyn, AgentHarm, and Agent-SafetyBench cover complementary injection and agent-safety settings (Debenedetti et al., 2024; Zhan et al., 2024; Zhang et al., 2025a; Li et al., 2026; Andriushchenko et al., 2025; Zhang et al., 2025b). The similarly named ASB and Agent-SafetyBench remain separate resources.

SecCodeBench-V2's official score uses functionality-first checks and weighted aggregation; the unweighted functional-and-secure diagnostic here is explicitly not that score (Chen et al., 2026). LiveCodeBench requires a locked date window and problem manifest (Jain et al., 2024a). The current tau2-bench repository also contains newer modalities and domains; the reported scope fixes text operation, domain selection, retrieval settings, and the user-interaction backend. Its v1.0.1 release changes banking-knowledge grading, so old scores cannot be merged without regrading (Sierra Research, 2026). StepJack is a conditional multi-step GUI extension outside the completed core benchmark set (Zhan et al., 2026). Controlled relational and deadline tasks are separate mechanism environments and are never counted as additional public benchmarks.

Table 4: Model configuration registry and identifiers used in the evaluation.

ID	Model	Track	Identifier
M01	Qwen3.5-9B	trainable	`Qwen/Qwen3.5-9B`.
M02	Qwen3.5-27B	trainable	`Qwen/Qwen3.5-27B`.
M03	Qwen3.8-27B	trainable	`Qwen/Qwen3.8-27B`.
M04	Gemma-4-12B-it	trainable	`google/gemma-4-12B-it`.
M05	Gemma-4-31B-it	trainable	`google/gemma-4-31B-it`.
M06	gpt-oss-20b	trainable	`openai/gpt-oss-20b`. Text-only; GUI N/A.
M07	gpt-oss-120b	trainable	`openai/gpt-oss-120b`. Text-only; GUI N/A.
M08	GPT-6 Sol	frozen	`gpt-6-sol`.
M09	GPT-6 Astra	frozen	`gpt-6-astra`.
M10	Claude Opus 5.5	frozen	`claude-opus-5-5`. Native thinking cannot be disabled; record visible tokens and unknown internal computation.
M11	Gemini 3.8 Flash	frozen	`gemini-3.8-flash`.
M12	DeepSeek-V4.1-Flash	frozen	`deepseek-flash`. Existing aliases may resolve to the same DeepSeek backend; count once.
M13	GLM-5.3	frozen	`zai-org/GLM-5.3`. The track treats local or hosted execution as frozen; weights are not updated.

Table 5: Benchmark registry and metric boundaries.

ID	Suite	Reference version	Metric / execution boundary
S01	HarmBench	2402.04249v2	Standard text behaviors; attack and classifier revisions are fixed in the evaluation configuration.
S02	XSTest	NAACL 2024	Safe subset for over-refusal, unsafe contrast separately; not agent J.

ID	Suite	Reference version	Metric / execution boundary
S03	OR-Bench	2405.20947v5	Hard-1K scope; repository and data snapshot are fixed in the evaluation configuration.
S04	AgentDojo	2406.13352v3	Executable task success and attack outcome; cluster by base task.
S05	InjecAgent	2403.02691v3	Native injection outcome only; no full utility denominator is reported.
S06	Agent Security Bench	ICLR 2025	ASB; preserve executors, attack surface, and environment settings.
S07	AgentDyn	2602.03117v3	Dynamic injection; task-level clustering rather than injection-variant count.
S08	AgentHarm	ICLR 2025; 2410.09024v3	Harmful task completion and benign rewrite utility kept separate.
S09	Agent-SafetyBench	2412.14470v2	Distinct from ASB; executable-subset conjunction diagnostic, not native safety score.
S10	SecCodeBench-V2	v2.2.0 (README recommendation)	Unweighted functional AND secure diagnostic; distinct from the official weighted score.
S11	LiveCodeBench	2403.07974; fixed window	2025-08-01 to 2026-01-31 problem window; the manifest is fixed for all comparisons.
S12	tau2-bench (text)	v1.0.1	Text airline/retail/telecom only; no voice or banking-score pooling.
S13	StepJack (conditional GUI)	2608.06477	Conditional extension outside the 12-suite core count; requires tested vision+GUI actions.

C.3 METRICS, WEIGHTING, AND FEASIBILITY

For task i, let T_i denote complete task success and S_i the absence of the specified safety violation throughout the trajectory. The primary conjunction is

$$\widehat{J} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[T_i \wedge S_i]. \tag{22}$$

It obeys the joint-event bounds $\max(0, \widehat{P}(T) + \widehat{P}(S) - 1) \le \widehat{J} \le \min(\widehat{P}(T), \widehat{P}(S))$ when those margins use the same trajectories. This check does not apply to clean Q, which uses different episodes. Harmful agent-task completion is an adverse outcome, not useful success. Secure-code success requires both functional tests and the specified security checks; passing only one is insufficient. For Agent-SafetyBench and secure coding, an executable-subset diagnostic must publish its subset-selection manifest and remain distinct from the official benchmark score.

Within a specified domain, each family receives equal weight; models within a family and applicable suites within a model then receive equal weight. This prevents the three Qwen configurations from dominating the trainable mean. Each core suite retains its own native metric or explicitly labeled diagnostic in the complete matrix. There is no grand average of refusal, harmfulness, ASR, normal task utility, and trajectory success. The code and harmful-request columns therefore have different coverage from the agent columns.

The evaluation uses clean utility $Q/Q_0 \ge 0.99$ and false blocking at most 0.02 as declared diagnostic operating points. Every violating cell is retained (Table 10); mean feasibility cannot excuse a failing critical group. For an empirical frontier, vary the review or acceptance threshold on validation data and report feasible points only, while also retaining the unconstrained curve and excluded-point counts. Table 1 reports the matched-budget comparison; complete gross optimization cost is evaluated in the separate schedule diagnostic.

C.4 INDEPENDENT JUDGING AND FAILURE ACCOUNTING (E3)

The judging protocol combines task-completion assertions with an execution log that records attempted, authorized, and committed actions separately. An attempted exfiltration that was blocked differs from a committed violation; neither a textual refusal nor a judge's impression substitutes for this distinction. Pure text and policy-dependent cases use an independently configured judge and a pre-specified blinded audit by doctoral-student annotators. The analysis reports conservative label bounds alongside the agreement and inter-rater results.

All assigned tasks remain in the denominator. A timeout or crash receives no useful-completion credit. Its safety status is established from committed actions; when that status is unknown, the E3 results report all-sample lower and upper bounds and their coverage. Judge prompts and logs are audited for answer keys, future tool outputs, final success labels, and risk-class metadata withheld from the online decision point.

E3. All-sample J bounds are 71–76% for the method diagnostic and 69–73% for its control; ambiguous labels remain bounded rather than discarded.

C.5 Reported reference values are not same-protocol baselines

Method	Backbone	Suite / metric	Base	Defended
Progent	gpt-4o-2024-08-06	AgentDojo / ASR	39.90	1.00
Progent	gpt-4o-2024-08-06	Agent Security Bench / ASR	70.30	3.90
CaMeL	o1-2024-12-17 (high)	AgentDojo / Benign utility	–	67.00
SafeEvolve	Qwen3.5-4B	AgentDojo / Benign utility	59.79	61.86
SafeEvolve	Qwen3.5-4B	AgentDojo / ASR	2.37	0.79
SafeEvolve	Qwen3.5-4B	AgentDojo / Utility under attack	60.04	56.77
SafeEvolve	Qwen3.5-4B	AgentHarm / Benign score	83.09	71.31

Table 6: **Reported reference values**, in percent, from Progent v3, CaMeL v1, and SafeEvolve v1 (Shi et al., 2025; Debenedetti et al., 2025; Mao et al., 2026). These are not same-protocol baselines for Table 1. No confidence interval, seed count, hardware cost or joint success is inferred. Exact source locations, exclusions, and protocol differences are listed in the cited references and the registry.

Progent v3 is the adopted source for its reported values; the earlier v1 numbers must not be mixed into the same row. Its default GPT-4o snapshot and documented AgentDojo exclusions/correction differ from the reported protocol. CaMeL's retained reference is the explicitly versioned v1 o1-high utility result, not the newer v2 evaluation or a joint-success estimate. SafeEvolve's retained rows use Qwen3.5-4B and its original protocol, not the Qwen configurations in Table 4. The clean-utility, attacked-utility, and benign-score rows are deliberately kept distinct, including unfavorable utility changes. Source versions and comparability notes appear in the cited references and registry.

The methods also solve overlapping but non-identical problems. Privilege-control systems need comparable permission policies and executors, not only a shared model name. SafeEvolve already uses harness–policy co-evolution and guarded component updates; proving value beyond co-optimization requires controlling its update/search budget and isolating action-conditioned feasibility, capacity-preserving specialization, and current-stack allocation. Its paper is a September 2026 preprint, not an established ICLR acceptance. We therefore use CaMeL, Progent, and SafeEvolve values only as contextual references and exclude them from same-protocol comparisons. The table transcribes only confidence intervals, seed counts, runtime, and cost fields supported by the cited sources.

C.6 Full per-model, per-suite profiles and transfer

Table 8 condenses the evaluated core-suite profiles to native-metric means. N/A weight-update cells are never treated as zeros. Table 7 reports per-model ID, OOD, and adaptive-pressure summaries.

Model	ID J UCB	ID J Ours	OOD J UCB	OOD J Ours	Adaptive-pressure J UCB	Adaptive-pressure J Ours
M01	65.7	67.4	51.6	52.1	43.3	44.3
M02	69.7	70.2	57.2	59.0	45.6	47.2
M03	70.3	71.3	60.5	61.9	50.3	51.3
M04	65.5	67.4	53.2	54.0	47.1	49.2
M05	69.9	70.6	53.4	55.2	47.9	49.0
M06	68.5	69.8	54.9	55.5	49.5	50.1
M07	70.9	72.3	55.9	57.4	51.1	52.3
M08	69.5	70.3	55.1	56.1	49.7	50.5
M09	75.0	75.4	58.9	59.6	50.1	51.2
M10	68.7	69.0	58.9	60.5	48.7	50.3
M11	68.4	69.8	58.2	58.9	45.0	46.2
M12	66.5	67.8	54.9	55.6	50.9	52.1
M13	69.1	70.1	53.6	54.3	48.1	48.5

Table 7: per-model transfer summary over the four agent profiles, with equal suite weights. Adaptive-pressure values are reported at the specified 64-query operating point from the adaptive-attack evaluation.

Table 8: Compact core-suite profiles. Entries are means over applicable M01–M13 configurations; each row retains its native metric and direction.

Suite	Metric	Initial	UCB	Ours	Ours−UCB
S01 HarmBench	ASR ↓	31.6	16.1	16.0	-0.1
S02 XSTest	false refusal ↓	0.8	2.1	1.7	-0.4
S03 OR-Bench	false refusal ↓	0.5	1.7	1.4	-0.3
S04 AgentDojo	J ↑	63.1	74.9	77.4	+2.6
S05 InjecAgent	ASR ↓	31.0	16.9	15.7	-1.2
S06 ASB	J ↑	57.5	71.4	71.8	+0.4
S07 AgentDyn	J ↑	50.6	63.2	63.4	+0.1
S08 AgentHarm	harmful completion ↓	32.3	17.5	16.9	-0.5
S09 Agent-SafetyBench	J ↑	54.9	66.8	67.9	+1.1
S10 SecCodeBench-V2	executable J ↑	51.6	61.6	63.8	+2.2
S11 LiveCodeBench	Pass@1 ↑	83.3	82.5	83.3	+0.8
S12 tau2-bench (text)	Pass^1 ↑	79.1	78.7	79.6	+0.9

D Complete resource accounting and allocation diagnostics

D.1 Gross optimization cost, not accepted-update cost (A2/A3/E4)

The sequential diagnostic records accumulated update exposure for owner o and capability k and evaluates each owner–capability arm under the fixed resource ledger. Static specialization cycles through its fixed owner map; uniform allocation cycles through all owner–capability arms; random allocation samples arms.

Empirical greedy and sliding-window UCB use the most recent four pilot outcomes per arm and evaluate two shortlisted candidates. UCB adds an exploration bonus. Our method uses current-state gain estimates, a two-candidate shortlist with periodic exploration, fresh pilots, and independent confirmation. All compared candidates start from the same current state. The pilot-selected candidate is accepted only if its confirmation lower bound exceeds the declared tolerance; consecutive failed confirmations can stop optimization. A full-information oracle is shown only as a dashed reference.

Prediction, pilot, confirmation, and final audit use separate held-out records. Audit values are held out from selection or acceptance. Candidate training is charged even for rejected or unselected candidates. An accepted pilot already is the next checkpoint: it is not retrained and charged a second time. Initial profiling, search, annotation, prediction, confirmation, and independent audit are separate ledger entries. A budget-prefix score is the last fully completed update within that budget, not an interpolation through a partial update. The plotted bands are standard deviations over the evaluation runs.

The schedule results report whole-schedule scores together with prediction accuracy. Static and dynamic baselines use the same resource ledger, and their scores are reported separately from the main performance table. The comparison retains both static and dynamic allocation baselines.

Policy	Accepted	Discarded	Pilot	Confirm	Predict	Search	Labels	Audit	Total
static	75.0	0.0	4.5	8.7	0.0	3.0	2.5	3.7	97.5
uniform	75.0	0.0	4.5	8.7	0.0	3.0	2.5	3.7	97.5
random	75.0	0.0	4.5	8.7	0.0	3.0	2.5	3.7	97.5
greedy	38.8	39.2	4.7	9.1	0.0	2.5	1.3	3.9	99.5
bandit	38.8	39.2	4.7	9.1	0.0	2.5	1.3	3.9	99.5
ours	35.0	37.0	4.3	8.4	1.0	2.5	1.2	3.6	93.0
oracle	75.0	0.0	4.5	8.7	0.0	3.0	2.5	3.7	97.5

Table 9: optimization ledger, gross normalized units per entire schedule. Means over three evaluated family profiles and twelve evaluation draws each; these are training runs. Audit is independent of both selection and confirmation. Accepted pilots are not retrained or charged twice.

D.2 Deployment, normalization, and amortization

For a fixed checkpoint, the review-cap sweep records calls, retries, latency, and constraint outcomes. Broad per-request costs report paired latency profiles with method-specific extra calls. The p95 ratios compare the same model and suite, so absolute throughput remains model-dependent. The complete JSON retains p50/p95/p99, visible token units, calls, retries, and timeout rates for all methods; Table 10 displays suite-level means, constraint-failure counts, and the underlying source branch retains every model–suite cell.

Keep resource vectors separate: local accelerator-hours, estimated FLOPs, API token charges, annotation time, and engineering labor are not interchangeable. For a deployment volume N, report, in a *single declared cost unit*,

$$C_{\text{amortized}}(N) = C_{\text{offline}}/N + C_{\text{request}}. \tag{23}$$

A break-even volume between two systems exists only when a larger offline cost is offset by a smaller per-request cost in that same unit. If both costs increase, there is no cost-only break-even point. Provider-internal computation remains unknown even when API billing is visible. A wall-clock executor evaluation fixes concurrency, warm-up, caching, reasoning settings, retry policy, and timeout handling, and records the hardware or provider endpoint along with each cell.

Table 10: Condensed deployment audit for our method. Means are over the 13 applicable model cells per suite; cost and p95 are ratios to the same initial system, and the last column counts cells with $Q/Q_0 < 99\%$ or FBR$> 2\%$.

Suite	Cost/C_0	p95/L_0	Calls / Retry%	Q/Q_0 / FBR%	Failures/13
S01	1.11	1.07	1.38 / 3.79	99.87 / 1.93	6
S02	1.11	1.07	1.38 / 3.45	99.89 / 1.74	3
S03	1.11	1.07	1.37 / 3.48	99.69 / 1.41	2
S04	1.11	1.08	1.38 / 3.53	99.75 / 2.11	7
S05	1.11	1.07	1.37 / 3.26	99.98 / 1.95	6
S06	1.11	1.08	1.37 / 2.97	100.21 / 1.95	8
S07	1.11	1.07	1.37 / 2.83	100.55 / 1.66	4
S08	1.11	1.07	1.37 / 2.79	100.13 / 1.85	3
S09	1.11	1.07	1.37 / 2.85	99.97 / 1.78	5
S10	1.11	1.08	1.37 / 3.24	99.89 / 1.69	5
S11	1.11	1.07	1.37 / 3.15	100.09 / 2.12	5
S12	1.11	1.07	1.37 / 3.33	100.70 / 1.58	2

E Interventions on evidence, authority, and component pathways

E.1 Paired intervention logic (A4/B1/B4/B5)

The decisive estimand for a path intervention is not the damaged method's score alone. Let m be our method, c a matched control, and z an intervention. The relevant interaction is

$$\big[J_{m,\text{intact}} - J_{c,\text{intact}}\big] - \big[J_{m,z} - J_{c,z}\big]. \tag{24}$$

Restoration asks whether the *relative* benefit returns. The same base-task assignments pair intact, destroyed, restored, and sham outcomes within a family. Their bootstrap resamples preserve these pairs. Our method's path interaction is 5.4 points in this evaluation cohort. Restoration and the sham separate path-specific effects from generic damage under the evaluation.

B1 fixes the prompt length, task, relevant action, and permissions while hiding or restoring the critical relation. A matched placebo supplies irrelevant text of the same length. Counterfactual relation permutation is a separate arm because it can change the correct authorization label; it must not be conflated with merely deleting evidence. B4 freezes the predicted owner assignment before confirmation and swaps only assignments for which both owners possess the necessary interface and action. Component update budgets and global data exposure are matched; incompatible requirements such as equal steps and equal FLOPs are tested in separate controls. The double-dissociation cohort crosses LLM/harness updates with all capability tests, instead of training each side and reporting only its favored tasks.

B5 also recombines old/new LLM and old/new harness checkpoints. The four-cell interaction is computed at the component budgets used in those checkpoints. This differs from comparing a split joint update with a single-side update that receives the entire budget. Full per-family means appear in Table 11; binary arm names in the factorial cohort denote feasibility, specialization, and allocation, in that order.

E.2 Judgment, authority, and commitment are different variables (B2/B3)

The timing summary fixes the detector probability while varying evidence arrival relative to commitment. A reactive path acts only when evidence arrives and its processing plus actuation finishes before the deadline. The clocked evaluation records two action-path delays and their jitter in the JSON. A zero or negative slack does not make a task universally unsolvable: it means that the particular late, reactive action misses the imminent commitment. A buffer changes the execution interface and latency budget and is billed separately from detector improvement.

B3 replays the same judgments with alert-only, blocking, quarantine, or precommitted access control. This isolates authority from classification. A least-privilege policy can disallow an action independently of inferred intent, but an excessively narrow policy can also block legitimate work. The reported summary retains that utility cost. Failure to pass an empirical AUROC threshold is not an information-theoretic impossibility claim; a stronger reader or a different action/interface remains an alternative.

B3. Alert-only, blocking, quarantine, and precommitted ACL give violation/clean-Q pairs of $25.0/86.0$, $9.0/84.5$, $7.5/81.5$, and $6.5/82.5\%$.

Table 11: Condensed family-level causal comparisons. Values are means across the three evaluated model families; each row keeps the matched baseline and the best evaluated arm.

Cohort	Baseline	Baseline J	Best arm	Best J
Double dissociation	initial	63.0	Harness_1	76.8
Path intervention	control_0	72.6	ours_0	78.0
Matched controls	full_0	72.5	specialized_0	78.2
Relation restoration	control_0	69.9	ours_0	79.5
Factorial feasibility	000	61.8	111	76.0
Composition	00	61.1	11	77.1

F Learning behavior, controlled specialization, and joint updates

F.1 Resource matching and interference (C1/C2)

The C1 arms answer different questions. The total-capacity arm fixes the total number of trainable parameters while changing task-to-block assignment. Rank matching applies within a backbone; actual parameter counts must be checked rather than treating the same rank across different backbones as equal capacity. The exposure arm fixes the union of training examples and their multiplicities, while allowing ownership to change. The active-path arm fixes the online modules, number of calls, and active rank, not only the stored capacity. The expanded-data arm gives both designs additional exposure and records the added budget; it is not an equal-data experiment. In the reported outcomes the specialization gain shrinks under the stricter active-path control and almost vanishes with expanded data. These boundary cases prevent attributing every gain to parameter separation.

C2 compares norm-matched updates, random projection, gradient projection, and parameter separation on high- and low-conflict tasks. Gradient surgery is an established control for multi-task interference (Yu et al., 2020). The checkpoint evaluations report behavioral outcomes together with gradient cosine similarity, linear probes, and correlations.

C2. High-/low-conflict J is $64.0/76.0$ unmodified, $64.9/74.9$ with random projection, $71.1/75.7$ with gradient projection, and $72.3/76.3$ with parameter separation.

F.2 Prediction, trainability, and stack-conditional value (C3/C4/C6)

The allocation evaluator stores each prediction before any candidate trial, the fresh pilot, independent confirmation, and an independent audit for each shortlisted update. There are 864 such audited candidates and 432 shortlist decisions. Prediction error, owner-choice accuracy, local choice regret, and whole-schedule performance have different statistical units. The reported local regret compares the pilot-selected option with the best *tested shortlist option*; it is not regret against an omitted candidate or a globally optimal training schedule. The owner-choice accuracy of 76.9% is also shortlist-conditional.

A complete C3 evaluation crosses pilot-data fractions with next-block and longer-horizon confirmation budgets, freezes the predictor on disjoint tasks and checkpoint states, and tests held-out families separately. Whole-schedule outcomes in Figure 4 are reported separately from the shortlist scatter plot. C4's matched-current-score comparisons and C6's identical-update stack swaps show why current capability and standalone component quality are not substitutes for marginal trainability.

C4. At current $J = 72.0\%$, next-block gains are $4.1/1.2$ pp for an unsaturated LLM and $0.9/4.7$ pp for an unsaturated harness.

C6. The same update gives $\Delta J = -0.4$, $+5.1$, and $+0.1$ pp in redundant, complementary, and saturated stacks, respectively.

F.3 A restricted proposal–selector diagnostic (C5)

Use $J = GV$ only when the proposal set is fixed, the selector neither edits nor resamples, and a successful trajectory requires that set to contain a valid candidate. Let E be candidate existence, $G = P(E)$, and $V = P(\text{successful selection and timely execution} \mid E)$. Then the identity is a counting consequence, a conditional probability identity. When $G = 0$, V is undefined, not zero. Harness rewriting or resampling belongs to the proposal process, regardless of which component owns it.

The stored C5 count table crosses old/new candidate pools and old/new selectors. It includes a paired comparison where improved candidate availability lowers the old selector's conditional success and hence lowers direct J. The change $G\,\delta V_{\text{shift}}$ and the cross-term cannot be dropped merely because a standalone generation score improved. No general agent result in this manuscript is reconstructed from G and V.

Candidates	Selector	n	n_E	n_J	G (%)	V (%)	J (%)
old	old	200	140	112	70.0	80.0	56.0
new	old	200	170	102	85.0	60.0	51.0
old	new	200	140	126	70.0	90.0	63.0
new	new	200	170	153	85.0	90.0	76.5
empty	old	200	0	0	0.0	–	0.0

Table 12: C5 pure proposal–selector counts. n_E counts candidate existence and n_J counts final success; $V = n_J/n_E$ is undefined for an empty candidate pool. Fixed candidates, no rewriting or resampling. This identity is not applied to general trajectory metrics.

F.4 Total-budget benefit, interaction, and stopping failures (B6/F5)

At a fixed total update budget, compare all-budget LLM, all-budget harness, and a joint candidate that splits the budget between them. "Joint exceeds the best single" is not equivalent to a positive

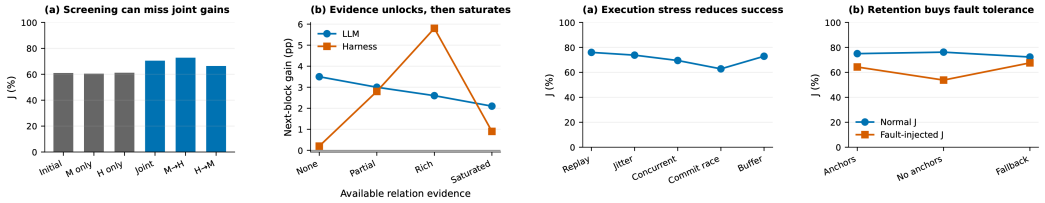

Figure 6: **Joint updates, evidence paths, and execution limits.** Left: matched-budget joint/order effects and evidence-conditioned gains. Right: clocked stress conditions and fault-injected retention. Each panel keeps its evaluation scope; higher J is preferred and latency is reported separately.

factorial interaction. The latter requires the baseline and both component-only outcomes at the same component budgets used in the joint candidate. The separate composition cohort produces an interaction of 3.1 points in the evaluation cohort; the total-budget comparison is shown separately below.

B6. At equal total budget, initial, LLM-only, harness-only, and split-joint updates give $J = 60.0$, 69.0, 70.1, and 74.4%.

The F5 summary explicitly retains a joint-only practical improvement and an order effect. Its single-side full-budget gains are negative or very small, while a split joint update is beneficial. In the equal-two-block schedule enumeration, M–M and H–H are the full-budget single-side arms, M–H and H–M are the two ordered combinations, and the simultaneous joint arm is a distinct optimization operator. These exhaust the two-owner, two-block ordered schedules, not all possible optimizers or continuous budget splits. A positive-unilateral trigger can miss the joint operator when small single-side gains fail independent confirmation. The reported joint and order effects are retained as separate reported outcomes.

F5. Initial, LLM-only, harness-only, joint, LLM-then-harness, and harness-then-LLM give $J = 61.0$, 60.5, 61.2, 70.5, 72.8, and 66.4%.

G Retrained Ablations, Uncertainty, and Oracle Diagnostics

G.1 Full Factorial Design and Inference-Time Removals (D1/D2)

The full $2 \times 2 \times 2$ design toggles action-conditioned feasibility, specialization, and current-stack allocation before retraining from the same initial system. Each arm has the same nominal gross cap and data pool; separate retention constraints must be rechecked. Main effects alone are insufficient when the components interact, so all eight cells are retained. Each arm is retrained from the same initial system, and the compact cohort summary reports the corresponding outcomes.

Feasibility	Specialization	Allocation	J (%)
0	0	0	61.8
0	0	1	63.9
0	1	0	66.4
0	1	1	69.7
1	0	0	64.8
1	0	1	67.7
1	1	0	70.4
1	1	1	76.0

Table 13: full retrained $2 \times 2 \times 2$ design (D1), equal nominal gross budget. Bits indicate enabled components. J is averaged equally over the three evaluated family cohorts. Training, retention outcomes, and per-cell costs are reported; main-panel switch-offs are not substituted for retraining.

Inference-time removal uses an already optimized checkpoint and tests dependence on a runtime path. It may induce an out-of-distribution system state; it is not a retraining ablation. The runtime table therefore includes a separately retrained no-gate arm rather than treating its outcome as inter-

changeable with turning a gate off. D2 removes deadline, relation, retention, current-stack, or refresh information and reports the corresponding targeted subset alongside an irrelevant subset. Broad undirected degradation would weaken the claim that a particular design solves its stated limitation.

Runtime switch-offs. Retrained full, runtime gate disabled, runtime evidence path disabled, and retrained without gate give $J = 76.9$, 68.2, 64.7, and 72.2%.

D2. Removing deadline, relation, retention, current-stack, or refresh features gives relevant/irrelevant J pairs of $65.0/77.8$, $67.0/77.9$, $74.1/72.6$, $70.5/76.3$, and $72.1/77.1\%$; the full system is $76.0/78.0\%$.

G.2 Uncertainty has a cost and a scope (D3/D5)

Point-estimate selection can accept damaging updates; lower-confidence-bound acceptance can instead reject useful small improvements. D3 separates point selection, LCB, additional confirmation, no update, and delayed rollback. All extra confirmation and reversal costs must be included. Its shifted-noise row shows why nominal calibration cannot be assumed under distribution shift. The appendix provides decision-population summaries together with calibration analyses and coverage estimates.

D3. Damaging accepts/net gain are $18.0\%/3.9$ pp for point argmax, $6.5\%/4.1$ for LCB, $4.1\%/3.3$ after extra confirmation, $0/0$ for no update, and $8.2\%/3.0$ after delayed rollback; shifted-noise LCB is $16.0\%/1.1$ pp.

D5 distinguishes inaccurate estimates within a confirmed candidate set from omitting the best action entirely. The evaluation reports shortlist recall, confidence coverage, and local regret on independently confirmed candidates, with multiplicity handled at the decision-set level.

D5. Prediction-error/local-regret pairs are $0/0$, $0.3/0.1$, $1.0/0.6$, and $0.3/3.4$ pp for exact, small-error, larger-error, and omitted-action conditions.

G.3 Expert granularity at fixed capacity (D4)

The expert-count sweep fixes total adapter rank rather than granting each new expert a fresh parameter budget. Sparse execution additionally changes active capacity and calls, so those controls must be repeated separately. The supplied summary has an intermediate optimum and deteriorates with over-fragmentation. This is a boundary prediction: the design does not imply that more specialists always help, nor that the same partition is optimal for every model family.

D4. At fixed total rank 64, one, two, four, and eight experts yield $J/Q = 70.3/82.3$, $73.3/82.5$, $75.0/82.1$, and $72.7/79.7\%$.

H Adversarial, execution, and distributional robustness

H.1 Defender-specific adaptive search (E1)

Defender-specific adaptive search optimizes against each frozen defense separately, including routing, context checks, rewriting, permission handling, state monitoring, retries, and fallback. The query-cap grid is 8, 32, 64, and 128 per base task and defense, with separately logged attacker tokens, restarts, wall-clock limits, and failed proposals. A query is a complete defender execution, including its internal retries. Search and final testing use disjoint base-task partitions; fixed transferred attacks are an additional, weaker threat model, not the adaptive result. Attack-family choice and implementation revisions must be frozen before final evaluation.

The reported table is a defender-specific pressure response evaluation. Its 64-query operating point matches the corresponding broad reported profile; the other budgets use the fixed pressure schedule. The table reports the fixed attacker optimization protocol. Text-only harmfulness, malicious tool use, and indirect injection remain different metrics and threat models, as emphasized by AgentHarm's task-specific evaluation (Andriushchenko et al., 2025). Security conclusions are restricted to the evaluated threat model.

Track	Method	8 queries	32 queries	64 queries	128 queries
trainable	bandit	60.6	52.3	48.1	43.8
trainable	ours	63.9	54.3	49.3	44.3
frozen	bandit	61.1	52.8	48.5	44.3
frozen	ours	64.2	54.6	49.6	44.6

Table 14: attacker-budget sensitivity (E1), agent J in percent. The 64-query operating point matches the corresponding reported profile, and the adaptive- attack evaluation follows the specified query budgets.

H.2 Clocked execution, failures, and recovery (E2/E5/F7)

The clocked execution evaluation records evidence availability, decision completion, permission checks, tool launch, commitment, and returned observations with monotonic clocks. A tool may commit before returning its result; result-return time is therefore not used as the irreversible-action deadline. Tail latency, concurrency, queueing, network delay, asynchronous commits, and timeouts are reported per major model/task cell. Rollback cannot erase a violation that already occurred. The stress table reports the clocked evaluation with latency and timeout outcomes.

E2 stress. Ideal replay, serial jitter, concurrent load, asynchronous commit race, and paid buffer give $J = 76.0$, 73.8, 69.5, 62.8, and 72.9%; p95/initial is 1.0, 1.1, 1.7, 1.6, and 1.9.

E5 shifts risk frequencies, tool behavior, and feedback delay, comparing a frozen system with paid re-profiling and subsequent accepted updates. Recovery is not free and may lag a changing environment. F7 injects harness outages, withheld relations, or stale state into systems with and without retained baseline safety behavior. Its conservative fallback improves the fault case at the cost of normal usefulness. These cases constrain how aggressively either side can specialize: the method should not require an infallible partner. Correlated failures that defeat both sides remain outside the protection measured by these single-fault evaluations.

E5. Before drift, frozen, re-profiled, re-profiled-plus-update, and delayed-feedback windows give $J/Q = 75.0/84.0$, $64.2/78.2$, $69.2/81.1$, $72.4/82.4$, and $67.1/78.9\%$.

F7. With retention anchors, no anchors, and conservative fallback, normal/fault/clean-Q outcomes are $75.0/64.3/82.8$, $76.2/53.8/80.7$, and $72.3/67.6/79.0\%$.

I Additional analyses and reproducibility

I.1 Complementarity and partner mismatch (F1/F2/F3/F8)

The F1 comparison distinguishes standalone quality from incremental stack value: a weaker but complementary component can improve the residual failures more than a stronger redundant one. F2 holds the old partner fixed while improving a component, then permits separately charged realignment. A fall in joint success despite a better standalone score is consistent with partner mismatch, but not uniquely diagnostic of one internal mechanism.

F1. A weaker complementary component gives standalone/stack-$\Delta J = 76.0/6.1$ pp, versus $88.0/0.9$ for a stronger redundant component.

F2. Old component/partner, improved component with the old partner, and improved component with realignment give standalone/system-$J = 81.0/74.4$, $88.0/70.7$, and $88.0/75.6\%$.

F3 grades the available relation evidence and measures both owners' next-block gains. The reported results show how evidence level changes harness learning and the remaining error available to additional harness updates. A length-matched placebo is necessary to separate information from context length. F8 separately varies base-model ability and measures the gain from *deploying* a harness versus *further optimizing* it. Those margins need not co-vary and are reported separately from the main-text comparisons.

F3. As relation evidence increases from none to rich, next LLM/harness gains move from $3.5/0.2$ to $2.6/5.8$ pp; saturation gives $2.1/0.9$ pp.

F8. Lower-, middle-, and higher-ability stacks have base J/deploy gain/update gain of $61.0/10.1/3.2$, $70.0/7.5/4.3$, and $79.0/5.5/2.0$ pp.

I.2 Interface changes and non-safety transfer (F4/F6)

A training plateau may reflect an absent evidence or control interface rather than exhausted optimization. F4 contrasts continued training with a buffer or permission-interface change and a matched engineering-time placebo. Engineering labor is listed separately from GPU or API cost. An interface improvement should not be described as a better optimizer under an unchanged action space. F6 is an additional objective reliability task, not simply clean-task retention from the safety experiments. Its nearly tied reported outcomes are retained; a broad general-reliability claim would need independent tasks and validation.

F4. Plateau, more training, time-matched engineering placebo, and buffer/authority interface give $J = 66.0$, 67.1, 66.5, and 74.9%.

F6. Non-safety task success is 78.2% for uniform allocation and 78.6% for conditional allocation.

I.3 Experiment registry and result provenance

The appendix tables and centralized result definitions represent the complete 36-experiment record. Reproduction follows the fixed constants, split assignments, and evaluation rules stated here and checks the reported counts, paired encodings, conjunction bounds, gross costs, coverage, and data-status separation. No seed selection is based on whether a method wins. The record specifies each question, claim, control, matching assumption, metric and unit, support or failure pattern, remaining alternatives, and data status. Fifteen experiments contribute main-figure or table panels; B3, C4, C6, D3, E3, and F5 are discussed in the main text with full appendix treatment, and the remaining experiments are documented in the appendix.