SCALE: CONTROLLING COMPOSITIONAL CAPABILITY GAINS IN BLACK-BOX LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

An assistant can withhold a complete answer while still releasing the pieces needed to reconstruct it. We study capability control over adaptive interactions, where the goal is to preserve permitted assistance while limiting an attacker's final task success. We propose SCALE, a controller that learns how much a candidate response changes task solvability given the current interaction history. Paired reconstruction attempts supervise before- and after-release solvability predictions, and an explicit difference loss emphasizes the candidate's contribution. Training pairs that hold the candidate fixed while varying earlier releases teach the auditor how this contribution depends on history. At deployment, SCALE screens candidates by predicted cumulative solvability and selects useful help with a penalty on positive predicted gain. Across Qwen3-8B and Llama-3.1-8B-Instruct on mathematics and code, SCALE retains 67.2–71.9% permitted-task utility at 20.6–21.8% attack success. Its utility exceeds that of the history-aware THRD baseline by 12.8 percentage points on average, while attack success differs by at most 0.6 points. Component studies examine the roles of difference fitting, gain-based selection, and fixed-candidate training pairs. We also derive a finite-bank calibration rule that selects episode-level policy mixtures under endpoint-risk and general-utility constraints, linking history-dependent response selection to guarantees on complete interactions.

1 INTRODUCTION

A model need not reveal a complete answer to make a restricted task solvable. A programming assistant can supply an algorithm, a boundary condition, and a stopping rule in separate responses, from which a recipient can assemble a working implementation. Each response may be acceptable in isolation. Refusing every related question avoids this failure, but at the cost of legitimate help such as explaining an interface or checking a user's own derivation. The challenge is to preserve such permitted operations while controlling what their accumulation enables.

The relevant unit of assessment is therefore the interaction, not the individual response. Consider a candidate response that explains a boundary condition. It may be of little use to a recipient who lacks the algorithm, yet it may complete a solution when earlier responses have already supplied the remaining components. The same candidate thus has different consequences in different histories. This observation raises two distinct questions: how solvable is the task after a release, and how much does the release itself change that solvability?

Capability control treats task accomplishment, rather than a refusal violation, as the endpoint (Siddiqui et al., 2026). Existing black-box interventions include prompt-based control in CAP and auxiliary behavioral divergence in CBD (Wang et al., 2026; Xie et al., 2026). History-aware defenses are the most relevant comparison: THRD incorporates turn-level risk, historical intent, and response facilitation (Ma et al., 2026), and Recast forecasts future failures from evolving compositional risk states (Lin et al., 2026). Our question is therefore narrower than whether history helps: *does learning the change in task solvability improve the selection of partial assistance beyond cumulative, history-based screening?*

We propose SCALE, a controller that wraps a frozen, text-only target LLM. Offline, a reconstruction probe attempts the same problem before and after a candidate is added to the visible history. The two outcomes supervise an auditor that predicts both cumulative solvability and its change. Training also includes pairs that hold the candidate fixed while varying earlier complementary information.

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.

At deployment, the controller screens candidate responses by predicted cumulative solvability and, among those that pass, selects useful help with a penalty on positive predicted gain. It receives neither reference answers nor private test feedback.

Local response selection and endpoint assessment operate at different scales. For the latter, we formulate a calibration procedure over complete interactive policies: simultaneous bounds over a frozen policy bank support utility-maximizing mixtures, with a single policy drawn for the entire episode. The result follows finite-family risk-control reasoning and accommodates adaptive requests from a declared finite set of attackers. Our experiments measure terminal attack success and permitted-task utility directly; the calibration result specifies how population-risk constraints can be imposed on a separately assessed deployment.

Our contributions are threefold. First, we operationalize response contribution through paired terminal reconstruction in a shared visible history. Second, we combine explicit difference fitting, fixed-candidate history comparisons, and gain-based selection to preserve partial help. Third, we establish a finite-bank endpoint calibration guarantee (Section 4) and evaluate the controller in four model/domain settings (Section 5). Relative to the history-aware THRD baseline, SCALE improves utility by 12.3–13.6 percentage points, while attack success differs by at most 0.6 points in absolute terms.

2 CAPABILITY CONTROL OVER COMPLETE INTERACTIONS

Access and scope. The target model M is frozen and exposes only sampled text. A policy P mediates all access to the target; neither target parameters nor token probabilities are available. The operator declares a restricted task family and a set of permitted operations. An episode comprises at most Q endpoint requests associated with a single server-side workspace identifier. Requests, released responses, and reset markers are recorded in an append-only control ledger. Chat resets preserve this ledger, the query count, and any initial policy draw; a new identifier starts a new episode. This design corresponds to an authenticated task workspace whose grouping the service can enforce, and the workspace constitutes the control boundary.

An attacker receives a problem x, fixed public metadata d, and the endpoint's responses. It may adapt its requests and combine all releases, but cannot access M outside the endpoint. After the interaction, it commits to a single final answer or program. For capability family j, the scorer $s_j(x, \widehat{y}) \in \{0, 1\}$ checks exact mathematical correctness or passage of the complete private test suite. Private references and test outcomes are available only for offline evaluation. The controller observes the capability specification and the actual requests, not a hidden benchmark statement supplied by the evaluation harness.

Endpoint risk and attacker competence. Let $\mathcal{D}_j$ be the declared distribution over independent semantic problem clusters. For a fixed complete attack algorithm a, define

$$R_{j,a}(P; d, Q) = \mathbb{E}_{x \sim \mathcal{D}_j,\, H_Q \sim (a,P)}[s_j(x, a_{\text{final}}(x, H_Q))]. \tag{1}$$

An attack algorithm comprises its model, prompts, search procedure, decoding, and non-endpoint compute budget. The declared attack set $\mathcal{A}$ is finite and frozen before calibration. It includes an ignore-endpoint algorithm a_0, whose success R_{j,a_0} is identical for every policy and therefore lower-bounds the worst-case risk over declared attacks. Any risk budget below this floor is infeasible. Assisted-minus-alone differences are informative paired diagnostics, but the certificate constrains absolute assisted success.

Permitted utility. A legitimate-user protocol is frozen separately from the attackers. The primary utility $U(P) \in [0, 1]$ is this protocol's success on operator-permitted tasks matched in topic and surface form to restricted requests, such as checking a supplied derivation step or identifying a local program error. Each permitted task has an explicit correctness criterion. Collateral loss on general tasks provides a separate deployment criterion:

$$D(P) = \mathbb{E}[u_{\text{gen}}(M) - u_{\text{gen}}(P)] \in [-1, 1]. \tag{2}$$

For binary general-task correctness, a tolerance $\varepsilon_{\text{gen}} = 0.03$ permits at most three percentage points of population accuracy loss. Section 4 incorporates this constraint into policy selection.

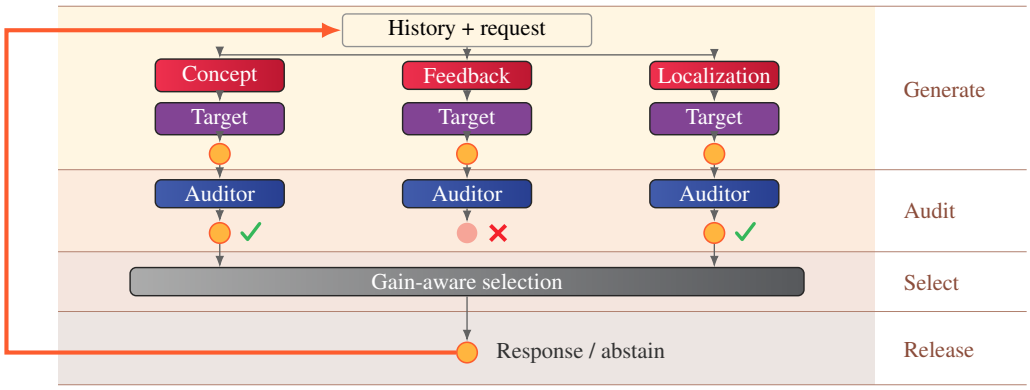

Figure 1: **SCALE: selecting partial help by its contribution.** For a gated request, the frozen target generates one candidate for each of three assistance plans. A shared auditor conditions on the visible history and current request to predict solvability and response gain. Candidates that pass the predicted-solvability and factuality checks enter utility-based selection with penalties on positive gain and cost. The controller releases one response, or abstains if no candidate passes. The orange loop appends only the request and the released response to the persistent history. Checkmarks and crosses illustrate screening outcomes.

Response contribution. We distinguish the interactive attackers $\mathcal{A}$ from a finite set $\mathcal{T}$ of offline terminal reconstruction probes. A probe r receives (x, h) and makes one final solution attempt with a fixed compute allowance and no further endpoint access. Define

$$V_r(x, h) = \mathbb{P}\{s_j(x, r(x, h)) = 1\}, \tag{3}$$

$$\Delta_r(x, h, y) = V_r(x, h \oplus y) - V_r(x, h), \tag{4}$$

where $h \oplus y$ appends candidate y as an assistant release to an otherwise identical visible record; the record includes the current user request in both branches. These probe-specific probabilities supervise local decisions. Gains may be negative, and complementary releases can change the value of later help. By contrast, Eq. (1) assesses the complete adaptive interaction.

Controlled composition diagnostic. For fragments $y_1, \ldots, y_k$, a probe receives either all fragments or a single fragment, in a fixed wrapper with the same problem statement and compute cap. Let h_K contain exactly the releases indexed by $K \subseteq \{1, \ldots, k\}$. The quantity

$$C_r = V_r(x, h_{\{1,\ldots,k\}}) - \max_{i \le k} V_r(x, h_{\{i\}}) \tag{5}$$

measures recovery beyond the best individual fragment. Key-fragment deletion and fixed-candidate history comparisons assess which earlier information contributes to reconstruction. Diagnostic construction and measurement use disjoint probe seeds.

3 SCALE: Selecting Help by Its Contribution

3.1 Candidates and Paired Reconstruction Supervision

A gate distinguishes restricted operations from semantically matched permitted operations, using the capability specification and the visible ledger. For a gated request, a level-conditioned auxiliary generator G_ϕ, which we call the *coarsener*, rewrites the request into three assistance plans: concept explanation, local-step feedback, and error localization. The target answers each plan under a fixed instruction envelope. Ungated requests receive a single ordinary candidate. Every candidate in a controlled policy goes through the release decision, so a gate miss does not bypass the auditor, and abstention is always available as a fallback. The target's parameters remain unchanged, and every internal target call is counted as cost.

For a training tuple (x, h, y) and probe r, we run S independent seed pairs with the same per-branch compute allowance. The paired scores are

$$z^-_{r,s} = s_j(x, r(x, h; \xi_s)), \qquad z^+_{r,s} = s_j(x, r(x, h \oplus y; \xi_s)). \tag{6}$$

Each seed coupling preserves the probe's specified marginal distribution. The means $\widehat{v}^-_r$ and $\widehat{v}^+_r$ estimate the two solvabilities, and $\widehat{\Delta}_r = \widehat{v}^+_r - \widehat{v}^-_r$ estimates their difference. Hidden references generate these offline labels but never enter as input features. Repeated seeds refine a label; they do not create new independent problem clusters.

A shared auditor $v_\theta(c, h, r) \in [0, 1]$ receives the capability specification c, the visible record, and the probe identifier. Writing $v^-_\theta = v_\theta(c, h, r)$ and $v^+_\theta = v_\theta(c, h \oplus y, r)$, we train it with

$$\mathcal{L}_{\text{audit}} = \sum_{r \in \mathcal{T}} \left[\text{BCE}(v^-_\theta, \widehat{v}^-_r) + \text{BCE}(v^+_\theta, \widehat{v}^+_r) + \lambda_\Delta \, \text{Huber}(v^+_\theta - v^-_\theta - \widehat{\Delta}_r) \right]. \tag{7}$$

The difference target is computed from the two outcome labels, so the additional term introduces no new labels; instead, it couples the two fitting errors and emphasizes the contrast used by the selector. Our factorial comparison varies λ_Δ while holding the labels fixed. Because the auditor conditions only on endpoint-visible information, its target is the reconstruction outcome predictable from that record. We use $S = 4$ seed pairs for labeling and $\lambda_\Delta = 1$ for auditor fitting.

Holding the candidate fixed. Training mixes ordinary attack trajectories with pairs (x, h_0, y) and (x, h_1, y). The two histories differ in whether they contain information complementary to y, while the current request and the candidate are identical. We construct them by retaining, replacing, or withholding earlier assistant fragments, never by inserting a hidden reference answer, and label them with actual probe outcomes. Both members of a pair remain in the same problem cluster. These examples directly train the dependence of Δ_r on the history, rather than on the candidate's vocabulary alone.

3.2 Selecting a release

For each candidate, define its predicted cumulative risk and positive gain:

$$\widehat{q}_\theta(h, y) = \max_{r \in \mathcal{T}} v_\theta(c, h \oplus y, r), \tag{8}$$

$$\widehat{g}_\theta(h, y) = \max_{r \in \mathcal{T}} \left[v_\theta(c, h \oplus y, r) - v_\theta(c, h, r) \right]_+. \tag{9}$$

Let $\widehat{u}(h, y)$ predict usefulness for permitted operations and $\widehat{f}(h, y)$ assess factual consistency. These auxiliary predictors are trained or configured on training data and frozen before calibration. With thresholds ρ and κ selected on development data, define the passing set

$$\mathcal{Y}_{\text{pass}}(h) = \{ y \in \mathcal{Y}(h) : \widehat{q}_\theta(h, y) \le \rho, \ \widehat{f}(h, y) \ge \kappa \}. \tag{10}$$

Among passing candidates, the controller releases

$$y^\star \in \arg \max_{y \in \mathcal{Y}_{\text{pass}}(h)} \{ \widehat{u}(h, y) - \lambda_g \widehat{g}_\theta(h, y) - \eta \, \text{cost}(h, y) \}. \tag{11}$$

Ties are broken by a fixed action order; if no candidate passes, the controller releases a fixed abstention. The cost term accounts for response length and any declared continuation cost; candidate-generation cost is incurred before selection and is charged regardless of which response is released. Negative predicted gains receive no reward, and only candidates passing the factual-consistency screen are eligible for release.

Cumulative solvability determines eligibility, whereas gain determines the ranking among eligible responses. With a single probe and positive gains, the subtracted before-release value is a candidate-independent constant, so the ranking coincides with that under a post-release solvability penalty. With multiple probes, each probe's own baseline is subtracted before the maximum is taken, and the two rules generally differ. The method thus combines an explicit contrast-fitting objective with a history-conditioned selection rule. Section 5 compares this rule with cumulative-screen-only selection.

Algorithm 1: Persistent-workspace response selection

Inputs: fixed policy bank $\mathcal{P}$, episode weights w, capability specification c, and request budget Q.
Initialize: draw $L \sim w$; set ledger $h = \varnothing$, counter $n = 0$, and overflow flag $o = 0$. Execute the settings of P_L throughout the episode.

1. While $n < Q$, receive a request; increment n, and append the request and any reset marker to h.
2. If h exceeds the policy's context limit, set $o = 1$. If $o = 1$, release the fixed abstention, append it to h, and proceed to the next request.
3. Apply the gate. Using the fixed retry rule, generate three assistance-plan candidates for a gated request, or one ordinary candidate otherwise.
4. For each reconstruction probe, predict solvability before and after each candidate, and compute $\widehat{q}_\theta$ and $\widehat{g}_\theta$ via Eq. (9).
5. Retain candidates that satisfy the cumulative-solvability and factuality thresholds. If none passes, select the fixed abstention; otherwise apply Eq. (11) with the fixed tie order.
6. Release the selected response and append it to h. Preserve (L, h, n, o) across chat resets.

Generator training. The coarsener is initialized on examples of the three permitted operations. On training histories, candidate plans are ranked by permitted-task utility minus a positive-reconstruction-gain penalty and cost; candidates that fail factuality checks are never preferred. Given a preferred plan z_{w} and a dispreferred plan z_{l} under capability specification c and visible history h, the update minimizes

$$\mathcal{L}_G = -\log \sigma\left(\beta_G \left[\log \frac{G_\phi(z_{\mathrm{w}} \mid c, h)}{G_{\mathrm{ref}}(z_{\mathrm{w}} \mid c, h)} - \log \frac{G_\phi(z_{\mathrm{l}} \mid c, h)}{G_{\mathrm{ref}}(z_{\mathrm{l}} \mid c, h)}\right]\right). \tag{12}$$

This is the standard preference objective of Rafailov et al. (2023); our specialization lies in the reconstruction-derived ranking, not in the optimizer. All auxiliary updates, checkpoints, thresholds, decoding settings, and stopping rules are frozen before calibration. Section 5 ablates the learned coarsener against a fixed-template coarsener and the explicit difference loss against an outcome-only auditor.

3.3 Complete policies and persistent state

At each request, the controller reads the persistent ledger, generates candidates, predicts before- and after-release solvability, applies the screen, and releases either one selected response or the fixed abstention. Only released text enters the attacker-visible record. A policy P_ℓ fixes every component above, including the context limit, gate threshold, candidate cap, refusal message, and backend retry cap. A finite bank $\mathcal{P} = \{P_1, \ldots, P_m\}$ collects development-selected settings. Pass-through and full-abstention policies can be included as reference policies; the ignore-endpoint attacker contributes its policy-independent success to the absolute endpoint risk of every policy.

For a mixture w, a policy index $L \sim w$ is drawn at episode creation, independently of the task and of the attacker's randomness, and P_L runs for the entire episode. Its internal decisions remain history-dependent. The draw, all released text, and the query counter persist across chat resets. If the full ledger exceeds the fixed context limit, the policy abstains for the remainder of the episode rather than silently dropping earlier releases. Each new workspace receives a fresh draw.

Algorithm 1 gives the online execution rule. If N requests reach candidate generation and N_{g} of them are gated, the three-plan architecture requires $N + 2N_{\mathrm{g}}$ target generations before retries, and hence at most $3Q$ per episode. Offline paired labeling requires $2S|\mathcal{T}|$ reconstruction attempts per training tuple, i.e., $8|\mathcal{T}|$ at $S = 4$. These counts distinguish endpoint requests, internal target generations, and reconstruction labels.

4 Endpoint Calibration for a Finite Policy Bank

For deployments with specified risk budgets, we give a selection rule based on independent observations of final task success. The guarantee follows from finite-family simultaneous bounds and the linearity of episode mixtures, in line with risk-control reasoning (Angelopoulos et al., 2021). The procedure operates on complete policies and leaves their internal response selectors unchanged.

Independent units and bounds. Fix a metadata/query-budget block (d, Q), and write $b = (j, a) \in \mathcal{B}$ for each declared family–attack constraint. On each of n_b independently sampled problem clusters, we run one complete episode with policy P_ℓ, producing $X_{ib\ell} \in \{0, 1\}$ and $k_{b\ell} = \sum_i X_{ib\ell}$. Scores of different policies on the same cluster may be dependent, but for each fixed policy and attacker, cluster scores are independent and identically distributed. With $\delta_R + \delta_U + \delta_D = \delta$ and $\zeta_R = \delta_R/(|\mathcal{B}|m)$, define the one-sided Clopper–Pearson upper bounds

$$\overline{R}_{b\ell} = B(k_{b\ell}, n_b; \zeta_R), \qquad B(k, n; \zeta) = \begin{cases} \mathrm{Beta}^{-1}(1 - \zeta; k + 1, n - k), & k < n, \\ 1, & k = n. \end{cases} \tag{13}$$

When a cluster's score averages multiple episodes, we instead apply a bounded-observation inequality rather than treating repeated episodes as independent Bernoulli samples. For n_U independent observations in $[0, 1]$, the utility lower bound is

$$\underline{U}_\ell = \max\{0, \widehat{U}_\ell - \sqrt{\log(m/\delta_U)/(2n_U)}\}. \tag{14}$$

For binary general-task correctness, let $F_{i\ell} = \mathbf{1}\{u_{\mathrm{gen},i}(M) = 1, u_{\mathrm{gen},i}(P_\ell) = 0\}$ indicate an error newly introduced by P_ℓ on paired task i. Writing $\mathbb{P}(1, 0)$ for the probability that M is correct and P_ℓ is not (and $\mathbb{P}(0, 1)$ for the reverse), we have $D(P_\ell) = \mathbb{P}(1, 0) - \mathbb{P}(0, 1) \le \mathbb{E}F_{i\ell}$, so

$$\overline{D}_\ell = B\Big(\sum_i F_{i\ell}, n_D; \delta_D/m\Big) \tag{15}$$

upper-bounds the net accuracy loss through the probability of a newly introduced error. For general scores in $[0, 1]$, the paired loss lies in $[-1, 1]$, and we instead use $\overline{D}_\ell = \min\{1, \widehat{D}_\ell + \sqrt{2\log(m/\delta_D)/n_D}\}$. The scoring mode and sample sizes are fixed before calibration.

Selection. For weights w in the simplex Δ_m, let

$$\mathcal{W}(\boldsymbol{\tau}) = \{w \in \Delta_m : \overline{R}_b^\top w \le \tau_b \ (b \in \mathcal{B}), \quad \overline{D}^\top w \le \varepsilon_{\mathrm{gen}}\}, \tag{16}$$

$$w^\star \in \arg\max_{w \in \mathcal{W}(\boldsymbol{\tau})} \underline{U}^\top w. \tag{17}$$

If $\mathcal{W}(\boldsymbol{\tau})$ is empty, no mixture over the bank is certified by these bounds at the specified budgets.

Theorem 1 (Uniform guarantee for episode mixtures)**.** *Condition on the training and development data. Suppose that the complete policy bank, scorers, legitimate-user protocols, metadata, and finite attacker set are fixed before calibration. Each attacker may adapt to observed responses but uses the same request and final-answer kernels for every mixture weight, and receives neither the calibration data nor the selected weights as input. Under the sampling assumptions above, with probability at least $1 - \delta$, simultaneously for all $w \in \Delta_m$,*

$$R_b(P_w) \le \overline{R}_b^\top w \quad (b \in \mathcal{B}), \qquad U(P_w) \ge \underline{U}^\top w, \qquad D(P_w) \le \overline{D}^\top w.$$

Consequently, every data-selected $w^\star \in \mathcal{W}(\boldsymbol{\tau})$ satisfies the stated risk and general-utility constraints.

Proof sketch. A union bound yields simultaneous validity over the finite bank. Conditional on the initial draw $L = \ell$, a fixed adaptive attacker interacts with P_ℓ alone, so each expected endpoint score is linear in w. Convex combinations preserve every inequality on the same event, uniformly over the simplex. The new-error construction additionally uses $D(P_\ell) \le \mathbb{E}F_{i\ell}$.

The theorem permits a history-dependent gate and candidate selector within each policy and allows dependence between turns. Its guarantee applies to the specified task distribution, attacker algorithms, and workspace boundary. An attacker re-optimized after deployment defines a new evaluation object: the attacker and deployment must then be frozen and the pair assessed on fresh calibration clusters. Appendix A gives the full argument, including the sampling units, confidence allocation, and finite-benchmark interpretation.

5 Evaluation

We evaluate the quality of permitted assistance jointly with an attacker's terminal task success. The main comparison covers two target models and two domains; a component study examines how difference fitting and response selection affect the controller.

Table 1: **Permitted-task utility and terminal attack success at $Q = 20$.** All entries are test-set percentages; higher utility U and lower attack success rate (ASR) are better. Each row is one evaluated configuration; shaded rows denote SCALE.

	Math		Code	
Method	$U \uparrow$	ASR$\downarrow$	$U \uparrow$	ASR$\downarrow$
Qwen3-8B				
Instruction	74.2	38.1	78.1	40.2
CAP (Wang et al., 2026)	61.3	24.7	64.2	26.8
CBD (Xie et al., 2026)	54.8	20.9	56.6	21.4
THRD (Ma et al., 2026)	57.1	20.4	59.4	21.8
Shared gate + abstention	48.9	17.6	50.2	17.1
SCALE	69.8	20.6	71.9	21.2
Llama-3.1-8B-Instruct				
Instruction	70.4	36.9	73.6	39.4
CAP (Wang et al., 2026)	58.1	23.9	60.8	28.1
CBD (Xie et al., 2026)	51.3	21.1	52.9	22.3
THRD (Ma et al., 2026)	53.6	20.8	55.8	21.6
Shared gate + abstention	45.1	18.2	46.8	17.9
SCALE	67.2	21.4	68.1	21.8

Tasks and access. We evaluate mathematical reasoning and code synthesis, using MATH and HumanEval+/MBPP+ as source benchmarks (Hendrycks et al., 2021; Chen et al., 2021; Liu et al., 2023). The protocol partitions source tasks into training, development, calibration, and final-test splits. A source task, together with its paraphrases, translations, fragments, and matched permitted operations, forms one cluster and stays within a single split. Qwen3-8B and Llama-3.1-8B-Instruct serve as frozen target models behind text-only adapters (Yang et al., 2025; Dubey et al., 2024). Reference answers and private tests are used only for offline scoring; the controller receives the capability specification, the requests, and the released history.

Attacks and endpoints. Attack protocols cover direct retries, paraphrase and translation, adaptive decomposition and aggregation, and an ignore-endpoint control. Each attack commits one final answer or program after at most $Q = 20$ endpoint requests; subrequests, blocked requests, and prompted resets all consume this budget. The attacker receives no private scoring feedback. Standalone and assisted conditions use matched non-endpoint compute; standalone success is 11.4% on mathematics and 16.8% on code. The attack success rate (ASR) is the fraction of episodes whose final answer is exactly correct (mathematics) or whose program passes the complete private test suite (code). Permitted-task utility measures correctness on operations such as checking a supplied derivation step, identifying a local error, and explaining a code interface; these requests are matched in topic and surface form to restricted ones.

Baselines. Instruction-only uses a fixed control instruction. CAP (Wang et al., 2026), CBD (Xie et al., 2026), and THRD (Ma et al., 2026) are adapted to the permitted operations through prompt optimization, auxiliary-model divergence-based routing, and temporal risk aggregation, respectively. A shared-gate control abstains on all gated requests. All methods use common data splits and matched caps on candidate count, although their auxiliary computation differs. The comparisons below therefore concern the resulting evaluated configurations, with THRD as the history-aware reference.

5.1 TERMINAL ATTACK SUCCESS AND PERMITTED UTILITY

Table 1 reports both test metrics for all six methods in each model/domain setting. SCALE achieves 67.2–71.9% utility at 20.6–21.8% attack success, whereas THRD achieves 53.6–59.4% utility at 20.4–21.8% attack success. The per-setting utility gains over THRD are 12.7, 12.5, 13.6, and 12.3 percentage points (equal-weight average: 12.8), and the corresponding attack-success changes are $+0.2$, -0.6, $+0.6$, and $+0.2$ points. The main empirical finding is thus a consistent increase in permitted assistance at comparable observed attack-success rates.

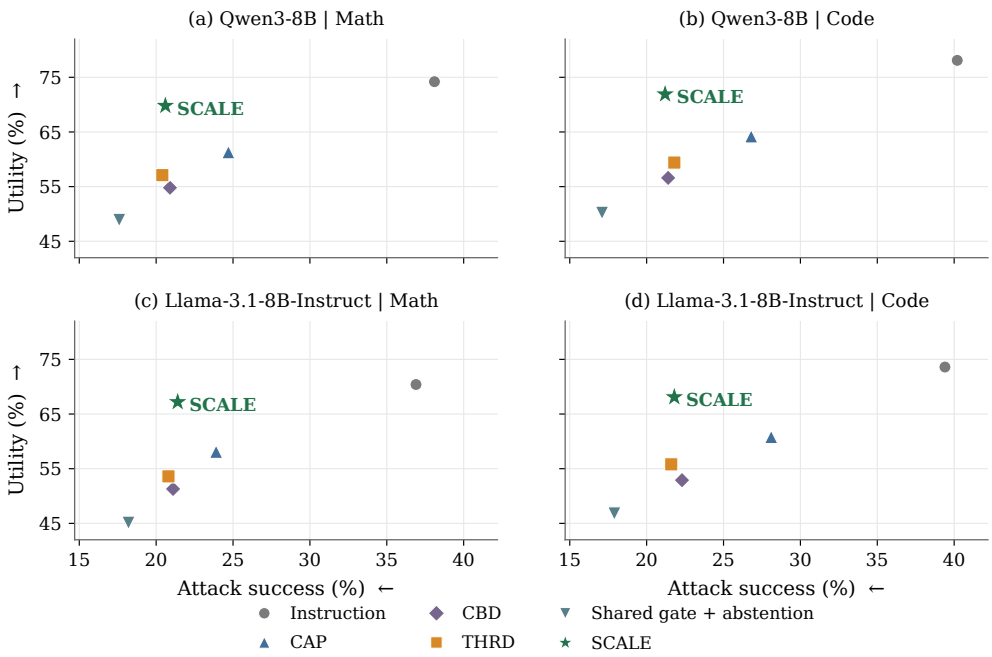

Figure 2: **Observed utility–attack-success tradeoffs.** Each marker is an evaluated configuration from Table 1; the upper left is preferred. Across all four settings, SCALE increases utility relative to THRD with small changes in observed attack success.

Figure 2 places this comparison within the broader tradeoff. Instruction-only retains the highest utility (70.4–78.1%) but has attack success of 36.9–40.2%. Relative to this control, SCALE lowers attack success by 15.5–19.0 points at a cost of 3.2–6.2 points in permitted utility. The shared gate with abstention reaches lower attack success (17.1–18.2%), but its utility falls to 45.1–50.2%; releasing selected partial help instead raises utility by 20.9–22.1 points over this control, at the cost of 3.0–4.1 points of additional attack success. CAP and CBD occupy intermediate operating points. The joint view shows how much permitted assistance each controller retains at its observed attack-success rate.

5.2 HISTORY DEPENDENCE AND COMPONENT BEHAVIOR

Reconstruction diagnostics. Full-record reconstruction succeeds on 56.6% of diagnostic problems; deleting a key fragment reduces success to 20.8%, a drop of 35.8 percentage points. In fixed-candidate history pairs, the candidate text and current request are unchanged while the earlier released information varies; across these pairs, the candidate's measured before/after gain differs between the two histories by 0.194 on average. Together, these diagnostics show that reconstruction is sensitive to complementary information already present in the history.

Auditor predictions. Adding the explicit difference loss reduces the before/after Brier score from 0.178 to 0.131 and the gain mean absolute error from 0.147 to 0.094. It also recovers the ordering of gains within history pairs in 81.2% of cases, compared with 61.4% for the outcome-only objective. Both auditors share the same backbone, visible inputs, training trajectories, and before/after labels; only the objective differs, placing more weight on the contrast between the two outcomes.

Difference fitting and selection. In the factorial comparison, adding the difference loss raises utility by 5.2 points with gain selection off and by 7.1 points with it on. Adding gain selection raises utility by 3.8 points without the difference loss and by 5.7 points with it. The full configuration is 10.9 points above the cumulative-screen-only configuration, with an arithmetic interaction of 1.9 points. These contrasts describe the learning and selection factors at their evaluated operating points.

Table 2: **Component study on Qwen3-8B/Math.** Entries are permitted-task utility (%) at the evaluated operating points. Left: difference loss × gain-based selection. Right: component replacements. These comparisons describe component sensitivity; Table 1 gives the joint utility and attack-success comparison for the full methods.

	Gain selection	
Difference loss	Off	On
Off	58.9	62.7
On	64.1	**69.8**

Variant	Utility
Without fixed-candidate pairs	66.4
Response-only auditor	63.1
Template coarsener	61.2
Single policy	64.4
Full SCALE	**69.8**

Removing fixed-candidate history pairs reduces utility from 69.8% to 66.4%. A response-only auditor yields 63.1%, and a template coarsener yields 61.2%. The single-policy alternative yields 64.4%. These results support retaining the history-dependent training examples, the learned assistance plans, and the bank-based controller together.

5.3 PERSISTENT STATE AND RESPONSE-LEVEL OUTCOMES

Within-workspace attack success is 20.6% without prompted resets and 21.1% with them. The controller preserves its ledger, query counter, and policy draw across these resets. At the request level, complete solutions are released on only 1.9% of restricted requests, while 8.4% of permitted requests are refused. A low complete-answer release rate can thus coexist with substantially higher terminal reconstruction success, because partial outputs remain available for aggregation. This gap underscores the need to evaluate complete interactions, not only individual responses.

6 DISCUSSION AND CONCLUSION

SCALE treats partial assistance as a history-dependent contribution to task solvability. Its auditor fits paired reconstruction outcomes and their difference, and its selector combines a cumulative-solvability screen with utility and gain. Across four model/domain settings, SCALE improves permitted-task utility over THRD by 12.8 percentage points on average, with attack-success changes between -0.6 and $+0.6$ points. The component study examines the roles of difference fitting, gain-based selection, and complementary-history training examples.

The empirical comparisons and the calibration theorem answer distinct questions. The former describe measured operating points; the latter provides a population-risk guarantee when complete policies are assessed on independent clusters against fixed attack algorithms. This separation allows the controller to be evaluated directly while retaining a precise route to risk-constrained deployment.

Limitations and future work. The present evaluation covers two 8B target models, mathematics and code, and twenty-request workspaces. Stronger reconstruction models, longer interactions, and histories pooled across workspaces are important next settings. Task-directed adaptations of PAIR and TAP (Chao et al., 2023; Mehrotra et al., 2024) could assess stronger search while retaining task completion as the endpoint (Souly et al., 2024). Comparisons with trajectory forecasting (Lin et al., 2026) and with tuned cumulative-risk penalties would further locate the contribution of difference-based selection within history-aware control.

AI USE STATEMENT

Generative AI tools assisted with manuscript editing and restructuring, checks of mathematical arguments and experimental reporting, literature verification, and LaTeX and figure preparation. The authors are responsible for the final manuscript and all reported results.

REFERENCES

Anastasios N. Angelopoulos, Stephen Bates, Emmanuel J. Candès, Michael I. Jordan, and Lihua Lei. Learn then test: Calibrating predictive algorithms to achieve risk control. *arXiv preprint arXiv:2110.01052*, 2021.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries. *arXiv preprint arXiv:2310.08419*, 2023.

Mark Chen, Jerry Tworek, Heewoo Jun, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Abhimanyu Dubey et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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

Shi Lin, Peng Qian, Dinghao Liu, Renjie Sun, Sifan Wu, Dezhang Kong, Chenpei Wang, and Xun Wang. Forecasting trajectory-level safety risks in black-box multi-turn interactions. *arXiv preprint arXiv:2607.26820*, 2026. URL https://arxiv.org/abs/2607.26820.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2305.01210.

Zhiqing Ma, Zhonghao Xu, Dong Yu, Chen Kang, Changliang Li, and Pengyuan Liu. THRD: A training-free multi-turn defense framework for jailbreak attacks on large language models. *arXiv preprint arXiv:2606.01738*, 2026. URL https://arxiv.org/abs/2606.01738.

Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. Tree of attacks: Jailbreaking black-box LLMs automatically. In *NeurIPS*, 2024. arXiv:2312.02119.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2305.18290.

Shoaib Ahmed Siddiqui, Eleni Triantafillou, David Krueger, and Adrian Weller. Position: Capability control should be a separate goal from alignment. *arXiv preprint arXiv:2602.05164*, 2026.

Alexandra Souly, Qingyuan Lu, Dillon Bowen, Tu Trinh, Elvis Hsieh, Sana Pandey, Pieter Abbeel, Justin Svegliato, Scott Emmons, Olivia Watkins, and Sam Toyer. A StrongREJECT for empty jailbreaks. *arXiv preprint arXiv:2402.10260*, 2024.

Zhaokun Wang, Jinyu Guo, Jingwen Pu, Hongli Pu, Meng Yang, Xunlei Chen, Jie Ou, Wenyi Li, Guangchun Luo, and Wenhong Tian. CAP: Controllable alignment prompting for unlearning in LLMs. *arXiv preprint arXiv:2604.21251*, 2026. URL https://arxiv.org/abs/2604.21251.

Zhiqiang Xie, Yijing Lin, Zhipeng Gao, and Dong In Kim. CBD: API-only LLM black-box unlearning through controlled behavioral divergence. *arXiv preprint arXiv:2606.27683*, 2026.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

A Proof and Scope of the Calibration Guarantee

Simultaneous bounds for the bank. Condition on training and development, so that all complete policies, attack algorithms, scoring rules, metadata, and sample sizes are fixed. For each constraint $b = (j, a)$ and policy ℓ, one episode on each independent cluster gives i.i.d. Bernoulli observations with mean $R_b(P_\ell)$. The one-sided Clopper–Pearson bound satisfies

$$\mathbb{P}\{R_b(P_\ell) > \overline{R}_{b\ell}\} \le \delta_R/(|\mathcal{B}|m).$$

A union bound therefore controls every risk row and policy with failure probability at most δ_R. The observations may be dependent *across* policies or attack constraints evaluated on the same cluster. For permitted-utility observations in $[0, 1]$, Hoeffding's inequality and a union bound give

$$\mathbb{P}\{\exists \ell : \ U(P_\ell) < \underline{U}_\ell\} \le \delta_U.$$

For binary general-task scores, write the paired outcomes as $A_i = u_{\mathrm{gen},i}(M)$ and $B_{i\ell} = u_{\mathrm{gen},i}(P_\ell)$. Then

$$D(P_\ell) = \mathbb{P}(A = 1, B_\ell = 0) - \mathbb{P}(A = 0, B_\ell = 1) \le \mathbb{P}(A = 1, B_\ell = 0).$$

The new-error indicators are independent Bernoulli observations across clusters. Applying the binomial bound with error probability δ_D/m and taking a union bound gives simultaneous upper bounds on $D(P_\ell)$. For nonbinary scores in $[0, 1]$, the paired difference has range two, so Hoeffding's inequality gives the radius $\sqrt{2\log(m/\delta_D)/n_D}$ used in the main text. Combining these events gives probability at least $1 - \delta_R - \delta_U - \delta_D = 1 - \delta$ that all bank inequalities hold.

From bank members to episode mixtures. For a fresh episode, the initial draw $L \sim w$ is independent of the task and all episode randomness. Conditional on $L = \ell$, the controller executes the complete policy P_ℓ. The attacker uses the same fixed algorithm and metadata that were used to assess that policy. An induction over requests identifies the conditional transcript distribution with direct interaction against P_ℓ: the same history produces the same conditional request distribution, followed by the same conditional response distribution. Consequently

$$R_b(P_w) = \sum_{\ell=1}^{m} w_\ell R_b(P_\ell),$$

$$U(P_w) = \sum_{\ell=1}^{m} w_\ell U(P_\ell), \qquad D(P_w) = \sum_{\ell=1}^{m} w_\ell D(P_\ell).$$

On the simultaneous bank event, multiplying the bank inequalities by $w_\ell \ge 0$ and summing proves all three bounds in Theorem 1 for every $w \in \Delta_m$. Uniformity over the simplex is a deterministic consequence of this finite event; it does not require an additional union bound over mixtures. The LP constraints then establish the claim for any data-selected feasible $w^\star$. □

Fixed attackers and public weights. An attacker may infer which policy is running from the released transcript; this behavior is included in its fixed adaptive algorithm. The preceding linearity argument requires that the algorithm's request and final-answer kernels remain the same for every bank member and every mixture. An attacker that additionally receives the selected weights, or is retrained after their publication, is a different evaluation object. To assess it, freeze the deployment, train the new attacker on development problems, freeze that attacker, and evaluate the resulting pair on fresh calibration data. This certifies the fixed pair, not arbitrary subsequent changes to the weights or attacker. The legitimate-user protocol has the same fixed-protocol requirement. Task-dependent routing can instead be assessed as a complete policy; its risk is not, in general, a constant-weight combination of the risks of its components.

Sampling unit and finite benchmarks. A semantic cluster contains its source problem and all derivatives, including paraphrases, translations, decompositions, matched permitted operations, and history pairs. Repeated probes improve a training label without increasing the number of independent source clusters. For calibration using multiple episodes per source cluster, let $\overline{X}_{ib\ell} \in [0, 1]$ be the within-cluster average and use, for example,

$$\overline{R}_{b\ell} = \min\left\{1, \frac{1}{n_b}\sum_{i=1}^{n_b} \overline{X}_{ib\ell} + \sqrt{\frac{\log(1/\zeta_R)}{2n_b}}\right\}.$$

Here n_b counts independent clusters, not episode repetitions. A benchmark name or deterministic split does not establish sampling from a broader problem population. A finite-pool claim must use its actual sampling scheme; sampling without replacement requires a corresponding finite-pool construction. Additional derivatives cannot justify an increased independent-cluster count. These requirements concern the scope of the population guarantee, separately from model familiarity with public tasks.

Confidence allocation and attainable budgets. For illustration, within a calibration block allocate $\delta = .05$ as $(\delta_R, \delta_U, \delta_D) = (.04, .005, .005)$. A bank with five declared constraints and $m = 14$ policies then gives $\zeta_R = .04/70$. Such blockwise 95% statements do not by themselves constitute a joint 95% claim across different methods, targets, or budgets; a joint claim requires allocation over those blocks as well. For zero successes the exact upper bound is $B(0, n; \zeta) = 1 - \zeta^{1/n}$, so even this best-case sample requires $n \geq \lceil \log \zeta / \log(1 - t) \rceil$ to certify a positive budget t. No finite all-zero sample certifies exactly zero population risk. The same calculation applies to newly introduced general-task errors. The new-error bound protects net accuracy conservatively because it does not offset introduced errors with improvements on other tasks.

B Training and Execution Details

Information available to each component. The target provides candidate text; terminal reconstruction probes provide offline labels; the auditor predicts from the visible record and capability specification. The offline probe additionally receives the task statement, a fixed reconstruction instruction, and its declared compute allowance. Both branches of a pair include the same current user request. Private references determine correctness labels but never enter the auditor's features or the online controller. Candidate text is treated as data, not as an instruction that can modify the capability specification.

Data collection and auditor fitting. Training starts from factually correct examples of concept explanation, local-step feedback, and error localization. A provisional controller collects complete training trajectories and candidate responses. Fixed-candidate pairs retain the current request and candidate while replacing or deleting earlier assistant fragments; measured reconstruction outcomes determine whether those histories are complementary. Both members of a pair remain in the same source cluster. Training uses $S = 4$ independent probe seed pairs per tuple and probe, with the same per-branch compute allowance and a coupling that preserves each branch's marginal distribution. Held-out diagnostics use separate probe seeds. Branches whose context would exceed the fixed input limit follow a common exclusion rule; unequal truncation would confound the before/after comparison.

The auditor is fitted with Eq. (7) and $\lambda_\Delta = 1.0$, with checkpoint selection on development clusters. The gain target $\widehat{\Delta}_r = \widehat{v}_r^+ - \widehat{v}_r^-$ is determined by the two outcome targets; it is not an additional independent label. The explicit difference loss emphasizes accurate estimation of this contrast. Accordingly, the gain-loss ablation retains both before/after records, both outcome targets, the architecture, and the training budget, and removes only the difference-loss term. The fixed-candidate-data ablation is separate: it replaces those pairs with the same number of ordinary-trajectory labels.

Training the assistance-plan generator. For factually acceptable candidates, preference scores combine measured permitted utility, positive reconstruction gain, and cost:

$$r_i = \widehat{u}_i - \beta \max_{r \in \mathcal{T}} [\widehat{\Delta}_{ri}]_+ - \eta \, \mathrm{cost}_i.$$

A development-fixed margin determines preference pairs; a factually invalid candidate cannot be preferred. The assistance-plan generator uses Eq. (12) in the main text. Target responses are sampled when constructing these preferences, and the target receives no gradient. All data collection, auxiliary updates, and development selection finish before calibration. The fixed-template variant replaces the learned plan generator.

Persistent execution and cost. A complete policy fixes its candidate cap, thresholds, decoding, refusal message, context limit, retry cap, and stopping rules. At workspace creation, the endpoint

draws one such policy and initializes the ledger and query counter. Every submitted request consumes a slot, including blocked requests and reset attempts. Chat resets preserve the ledger, counter, and initial draw. Ledger overflow causes abstention for the remainder of the episode. Missing candidates follow the fixed retry rule rather than being regenerated until a passing answer appears. The ledger records the request and actually released response; unreleased candidates remain private. Refusals remain part of the transcript assessed by the final attacker.

Candidate-generation calls are charged even when their responses are rejected. Target calls and auxiliary calls are separate quantities: the gate, coarsener, and auditor do not become target calls merely because they support one endpoint response. The naive calibration workload is $m \sum_{b \in \mathcal{B}} n_b$ episode-score evaluations per bank. A policy-independent ignore-endpoint control may reuse one evaluation when its distribution and confidence construction are unchanged.

C Interpreting the Component and Composition Analyses

Factorial contrasts. Let $U_{ab} \in [0, 1]$ denote utility, with a indicating the difference loss and b indicating gain-based selection. Converting the percentages in Table 2 to proportions gives

$$U_{11} - U_{10} - U_{01} + U_{00} = .698 - .641 - .627 + .589 = .019, \quad \text{or } 1.9 \text{ percentage points.}$$

The loss factor changes the training objective while retaining both before/after labels. The selection factor changes how candidates passing the cumulative screen are ranked. These point contrasts characterize the evaluated configurations; uncertainty in the interaction requires paired source-cluster observations.

Composition statistic. Equation (5) takes the best individual fragment within each problem. Therefore

$$\mathbb{E}C_r = \mathbb{E}V_{\text{full}} - \mathbb{E}[\max_i V_i],$$

whereas subtracting the largest population-average singleton value produces a different quantity. A plug-in maximum of noisy fragment estimates has selection bias. Independent construction and measurement seeds, together with source-cluster aggregation, keep the diagnostic aligned with its defined unit of analysis.

Workspace scope. Persistent state groups the requests within one workspace. To see why new workspaces change the evaluation object, suppose a fixed task is solved whenever a permissive policy is drawn with probability p and is never solved otherwise. Across K independent workspaces, at least one success has probability $1 - (1 - p)^K$. With task-dependent success probabilities p_x, the population quantity is $\mathbb{E}_x[1 - (1 - p_x)^K]$. Pooling records for a new reconstruction defines a further endpoint. Its assessment must include the pooled information and the total request and reconstruction budgets.