Individual Credit, Joint Effects: Understanding Credit Assignment in Agentic RL

Anonymous authors
Paper under double-blind review

Abstract

Credit-assignment methods give each action of an agent its own learning signal, and most recent work improves how these signals are constructed. How the signals change the policy once they enter training, which should inform their design, remains poorly understood. In GiGPO training on a controlled agentic task, 37.5% of actions with nonzero credit change probability against it. To trace why, we decompose each Adam update into contributions from individual current and past actions, which we call sources. After the first update, removing the four most strongly opposing sources restores the credit direction in 75.8% of reversals with count- and norm-matched controls, against 16.7% for other opposing sources. From the third update on, most reversals also require removing historical sources, partly because past gradients gain weight in Adam's first moment. Compared with candidate actions, the removed sources more often perform the same kind of operation as the target when their credit is opposite, and share one of its immediate effects when their credit has the same sign. A credit signal may thus reach related actions in these two ways, within an update and, through momentum, in later ones, which raises the question of how much history the optimizer should retain. We also test a history-retention adjustment that uses only existing optimizer state: with GiGPO and GRPO on ALFWorld and WebShop, uniformly attenuating retained momentum has no consistent effect on final success, consistent with attenuation scaling both helpful and harmful history equally.

1 Introduction

Reinforcement learning is increasingly used to train language-model agents for multi-turn tasks, from interactive environments to web navigation (Zhou et al., 2024; Wang et al., 2025b; Wei et al., 2025). Such tasks rarely provide step-level rewards: the reward usually scores an entire episode, so a successful episode may contain redundant actions and a failed one may contain necessary preparation. Credit assignment converts this episode-level feedback into a learning weight for each action, and the policy is optimized with these weights (Feng et al., 2025; Luo et al., 2025).

Most existing methods focus on how these weights are constructed. They compare an outcome with others sampled for the same prompt (Shao et al., 2024; Ahmadian et al., 2024), compare individual steps against references built from matching observations, histories, or trajectory graphs (Feng et al., 2025; He et al., 2026; Li et al., 2026), or learn step-level feedback (Wang et al., 2025a; Liu et al., 2026; Xi et al., 2026), and they are typically evaluated by task success and component ablations. Yet each method produces one coefficient per action, and what that coefficient does to the policy is determined only when it enters an update together with all the others.

Such an update does not apply each signal in isolation: training on one response changes the model's outputs on others through shared parameters (Ren & Sutherland, 2025). In critic-free RL, this coupling can cancel what successful and failed trajectories share (Cheng et al., 2026) or keep the likelihood of correct responses from rising (Deng et al., 2025), and in agents it can carry reinforcement into contexts where an action is inappropriate (Liu et al., 2025). Which experiences act on a given action, and through which relations, remains unclear. To design credit assignment beyond component ablations, a prior question is how credit coefficients combine with other current and past signals to form an update and change specific action probabilities.

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.

Equation (1) implies that positive credit should raise an action's probability and negative credit should lower it. If this always held, the coefficients alone would describe what credit does in training. In GiGPO training on a controlled text-based manipulation task, however, 37.5% of nonzero-credit actions move against their credit across 106 updates, with reversals in every update (Section 2.3). Similar advantage–probability discrepancies have been reported in critic-free RL and attributed to token coupling within a batch (Cheng et al., 2026). Under Adam, each update also carries earlier batches through momentum.

Expanding Adam's first moment writes each update as a weighted sum of the gradients of all actions observed so far. The change in an action's probability then depends jointly on the other actions in its batch and on earlier actions retained in momentum. Because an action's own contribution always locally supports its credit direction, a reversal implies, to first order, that the combined contributions of these other records outweigh it. We call them sources. For each reversed action, we remove sources in order of how strongly they oppose it until its credit direction is restored, and call the removed prefix its recovering set. From the second update on, the top-ranked sources restore the credit direction far more often than count- and norm-matched controls, although their recovery rate declines over training. Most recovering sets that mix current and historical sources also fail when only their current members are removed, so records from earlier batches contribute to these reversals alongside current ones.

To characterize these sources, we audit the members of recovering sets with environment rules and blinded judgments from a frozen language model. Two relations are consistently more common among them than among candidate actions. Members whose credit is opposite to the target's usually perform the same kind of operation, often in a different state. At the first update, for example, the credit assigned to `open box`, which gives access to a book, increased the probability of a penalized `open basket` in another trajectory, where the basket serves no remaining goal. Members with the same credit sign usually share an immediate effect with the target. Penalizing `drop pen` increased the probability of a penalized `put pen in basket`, and both actions free the hand. At the first update, many such members are alternatives that the policy could have chosen instead, and their opposed gradients are consistent with competition.

Either relation may carry a credit signal to other actions, and momentum extends this reach to later updates. How much history to retain thus becomes a question for optimizer design. Section 4 tests one such adjustment, which attenuates all retained momentum uniformly because removing only the opposing sources would require per-action gradients.

Our contributions are: (1) a record-level expansion of Adam updates in GiGPO training into per-action sources, including those carried by momentum; (2) matched deletion controls and blinded audits that locate current and historical sources of probability reversals and identify two recurring relations between sources and the actions they affect; and (3) a test of history retention computed without per-action gradients, with GiGPO and GRPO on ALFWorld and WebShop, showing no consistent benefit from uniform attenuation.

Section 2 sets up the controlled GiGPO training and measures whether action probabilities follow their credit. Section 3 decomposes each update into sources, tests them by deletion, and characterizes the experiences behind them. Section 4 tests history retention on ALFWorld and WebShop, Section 5 discusses related work, and Section 6 concludes.

2 Study Setting and a Direct Check

2.1 Action Credit and Its Role in Training

At update t, batch B_t contains executed action records: input history $h_{t,i}$, complete response $a_{t,i}$, feedback, and originating trajectory. Repeated commands are kept as separate records, because the credit of each occurrence enters the loss. With total credit $A_{t,i}$, trainable LoRA parameters θ, and policy π_θ under the legal-action mask, the loss is (Williams, 1992)

$$L_t(\theta) = -\frac{1}{Z_t} \sum_{i \in B_t} A_{t,i} \log \pi_\theta(a_{t,i} \mid h_{t,i}). \tag{1}$$

The log-probability of an action sums over its valid response tokens, and Z_t counts these tokens across the batch. Credit applies to every response token, and $A_{t,i}$ and Z_t are held fixed during

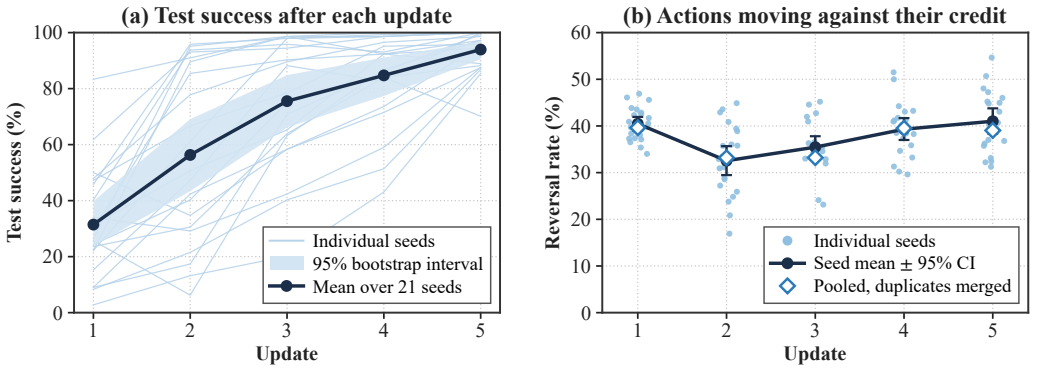


Figure 1: Task learning and credit directions. **a**: Success on 144 fixed test tasks after each update for 21 seeds (thin lines), their mean (thick line), and a 95% bootstrap interval over seeds (band). **b**: Fraction of nonzero-credit actions whose probability moves against their credit, per seed (dots) and as a seed mean with 95% bootstrap interval; 22 seeds at update 1 and 21 thereafter. Diamonds give the pooled rate after merging identical loss terms, which changes the pooled rate by at most 2 percentage points.

differentiation. Through its own loss term, positive credit tends to raise an action's probability and negative credit tends to lower it. This is the expectation we test in Section 2.3.

2.2 Traceable GiGPO Training

Our text-based manipulation task has six objects, three containers, and three placement goals. Object locations, the held object, container states, and completed goals all vary across tasks, so the same command can advance a goal in one task, merely prepare another action in a second, and undo progress in a third. We log the full state and feedback at every step, so each action can later be read in its original context.

GiGPO combines trajectory-outcome and observation-matched step comparisons (Feng et al., 2025): $A_{t,i} = E_{t,\xi(i)} + S_{t,i}$, where E and S are the episode and step terms for action i in trajectory $\xi(i)$. We keep its native construction and log both terms together with the reference records they are computed from.

We train Qwen3.5-0.8B with LoRA (rank and alpha 8) (Qwen Team, 2026a; Hu et al., 2022) on batches of three tasks with eight trajectories each, using native GiGPO credit ($\gamma = 0.95$) and Adam with learning rate 3×10^{-4}. Sampling, credit computation, and loss accumulation use the same pre-update parameters. A single Adam update (Kingma & Ba, 2015) follows, with no extra losses, gradient clipping, or weight decay. Parameters and optimizer state carry over from one update to the next. These conditions make the action losses the only signal passed to Adam, so the decomposition in Section 3 has no other terms apart from a small numerical residual. Because inputs, credit, and checkpoints are saved, every action's gradient can later be recomputed at the parameters under which it was taken.

Measurements cover 106 updates and 28,363 actions: 21 seeds with five updates each, plus the first update of one more seed. Mean test success across the 21 complete runs rises from 31.4% after update 1 to 93.9% after update 5, with fluctuations in individual seeds (Figure 1a). The analysis below concerns runs that are learning the task.

2.3 Does an Action's Credit Predict Its Probability Change?

We measure action log-probability before and after each update, holding the input history, response, and state-dependent legal-action mask fixed:

$$\begin{aligned} \Delta\ell_{t,i} &= \log \pi_{\theta_t}(a_{t,i} \mid h_{t,i}) - \log \pi_{\theta_{t-1}}(a_{t,i} \mid h_{t,i}), \\ r_{t,i} &= \operatorname{sgn}(A_{t,i})\Delta\ell_{t,i}. \end{aligned} \tag{2}$$

For nonzero credit, $r_{t,i} > 0$ follows the credit direction, $r_{t,i} < 0$ reverses it, and $r_{t,i} = 0$ is counted separately. We compare only directions of change, regardless of whether an action is good or bad. Reversal rates are computed over nonzero-credit actions.

Of 28,363 nonzero-credit records, 10,636 reverse (37.5%), and reversals occur in all 106 updates (Figure 1b). One might suspect that repeated records, each counted separately, inflate this rate. Merging identical input/response/mask records within each update and summing their credit still leaves 7,598 reversals among 20,581 nonzero-credit loss terms (36.9%). Duplicates do not account for the reversals, and an action's own credit does not reliably predict how its probability will change. To understand what credit does in training, we examine the other signals that form the update.

3 How Credit Signals Jointly Change Specific Actions

An action's own contribution to an update always locally supports its credit direction, as shown below. To first order, a reversal therefore implies that the combined contributions of other records, from the current batch and, through Adam's momentum, from earlier batches, outweigh it. We identify these records by writing out a single update record by record.

3.1 Tracing an Update to Current and Historical Actions

For source record (s, j), let $b_{s,j} = \nabla_\theta \log \pi_\theta(a_{s,j} \mid h_{s,j})|_{\theta_{s-1}}$ be its gradient at the original parameters. With Adam's actual second moment v_t, set $D_t = \mathrm{diag}((\sqrt{v_t/(1-\beta_2^t)} + \epsilon)^{-1})$ (Kingma & Ba, 2015). Expanding the first moment from $m_0 = 0$ gives the displacement $\Delta\theta_t = \theta_t - \theta_{t-1}$:

$$\Delta\theta_t = \sum_{s=1}^{t} \sum_{j \in B_s} U_{s,j}^{t}, \qquad U_{s,j}^{t} = \frac{\eta_t (1-\beta_1)\beta_1^{t-s}}{(1-\beta_1^t) Z_s} A_{s,j} D_t b_{s,j}. \tag{3}$$

We call each record (s, j) a *source* of update t, current for $s = t$ and historical for $s < t$, and $U_{s,j}^{t}$ is its contribution to the displacement, carrying its original action and credit. Historical gradients keep their original values, and all sources share the current D_t rather than passing through separate Adam transformations. Appendix C gives the optimizer equations, derivation, and numerical checks.

For a current target i with nonzero credit and $\|b_{t,i}\|_2 > 0$, define

$$d_{t,i} = \mathrm{sgn}(A_{t,i}) \frac{b_{t,i}}{\|b_{t,i}\|_2}, \qquad q_{i \leftarrow (s,j)}^{t} = d_{t,i}^{\top} U_{s,j}^{t}. \tag{4}$$

Positive q locally supports the target's credit direction, and negative q opposes it. Because D_t is a positive diagonal matrix, the target's own contribution always supports it, whereas other records can act in either direction. Excluding the target's own current record, we rank the sources with negative q. Since q is only a local quantity, we next measure their effects with forward passes.

3.2 Testing the Effects of Selected Sources

For a reversed target ($r_{t,i} < 0$), let $S_{t,i}(k)$ contain the first k negative-q sources in increasing order of q. We subtract their contributions from the realized post-update parameters and score the original input, response, and mask:

$$\begin{aligned} \theta_{t,i}^{(-k)} &= \theta_t - \sum_{(s,j) \in S_{t,i}(k)} U_{s,j}^{t}, \\ r_{t,i}^{(-k)} &= \mathrm{sgn}(A_{t,i}) \left[\log \pi_{\theta_{t,i}^{(-k)}}(a_{t,i} \mid h_{t,i}) - \log \pi_{\theta_{t-1}}(a_{t,i} \mid h_{t,i}) \right]. \end{aligned} \tag{5}$$

$r_{t,i}^{(-k)} > 0$ means the credit direction is recovered. Unlike retraining without the underlying data, the test keeps source vectors and the Adam scaling of that update fixed, and so measures what these sources contributed to the update that took place.

Removing a few sources restores many targets. We test $k = 1, \ldots, 32$ and record first recovery at $k_{t,i}^{*}$. We call the prefix $S_{t,i}(k_{t,i}^{*})$ the target's recovering set. Of 10,636 reversed targets, 47.3% recover

after one deletion, 80.6% within four, and 99.4% within 32, leaving 63 unrecovered (Figure 2a). These rates pool all five updates, including the first, where recovery is easiest (see below). Note that first recovery follows this one fixed ranking, ; a smaller set or different combination might also restore the target.

Ranked sources outperform count- and norm-matched controls. One possible confound is that a sufficiently large deletion could flip any target. At fixed $k = 1, 4, 8$, we compare the ranked prefix with two control sets, drawn from other opposing sources or from supporting sources. Each control has exactly k sources, and the norm of its summed displacement is within $\pm 10\%$ of the prefix's, without rescaling. We compare only targets for which both controls can be matched, and fix the matching before evaluating recovery.

For the 2,830 common targets at $k = 4$, removing the strongest opposing sources restores 85.1% of targets, compared with 36.9% for other opposing sources and 4.0% for supporting ones (Appendix D). Weighting the 22 seeds equally gives 85.8% (95% bootstrap interval 82.2–89.1), 38.3%, and 4.1%. The ordering holds in every seed and at the other two values of k. At matched deletion size, recovery thus depends on which sources are removed.

The first update, which supplies 1,182 of these targets, gives a weaker contrast. With $m_0 = v_0 = 0$, bias correction makes Adam's first step approximately $-\eta_1 \operatorname{sgn}(g_1)$, and $D_1 \approx |g_1|^{-1}$ magnifies single contributions wherever the batch gradient nearly cancels. Even other opposing sets recover 65.1% of targets there, against 98.0% for the strongest. Over updates 2–5, the 1,648 common targets (20.1% of the reversals at those updates) give 75.8%, 16.7%, and 0.6%, a sharper separation.

The ordering holds at every update, but recovery by the strongest sources falls as training proceeds, to 54.1% at update 5 (Figure 2b). The matched comparison covers 19.3%, 26.6%, and 12.0% of all reversed targets at the three values of k.

Median signed log-probability changes confirm the same ordering, and at $k \geq 4$ the deleted vectors exceed the actual update norm, so the test probes finite rather than local effects.

Many recovering sets reach back to earlier batches. The sources that matter are not confined to the current batch. Over updates 2–5, 54% of reversals have recovering sets with historical members, and for 20% every member is historical. By update, sets with historical members account for 22.9%, 59.7%, 69.6%, and 63.4% of reversals (Figure 2c). Up to update 4 this follows the weight of past gradients in Adam's first moment, $(\beta_1 - \beta_1^t)/(1 - \beta_1^t)$, which is 47%, 63%, 71%, and 76% at these updates, so part of the rise comes from optimizer mechanics rather than from the particular experiences. Deleting only the current members of mixed sets rarely restores the target (121 of 2,764), but this is expected for the shortest recovering prefix and does not show that the historical members matter more.

3.3 From Source Contributions to Training Experiences

Deletion establishes what these sources do to the target but not what experiences they represent. We audit the members of recovering sets, called selected sources below, to characterize the source and target actions, separating current from historical sources and same-sign from opposite-sign credit. Comparisons across updates use 19 common seeds. The audit combines environment rules with frozen, blinded judgments from Qwen3.8-27B (Qwen Team, 2026b), which sees the task, the state before the action, the action, and the preceding experience, but never credit, q, or recovery outcomes. Appendix E gives the rules, scope, and disagreements. Because the audit reads only sources that deletion has already located, it yields action relations that can be verified record by record.

Equation (4) motivates the split by credit sign:

$$\operatorname{sgn}\left(q^t_{i \leftarrow (s,j)}\right) = \operatorname{sgn}(A_{t,i} A_{s,j}) \operatorname{sgn}\left(b_{t,i}^\top D_t b_{s,j}\right).$$

For an opposing source, opposite-sign credit implies a positive gradient interaction (gradients aligned under Adam's scaling), and same-sign credit a negative one (gradients opposed). We ask which action relations accompany each case: shared operation type, shared immediate effects, similar states, and whether each action was available as an alternative in the other's state. Environment rules determine operation type and shared effects; frozen, blinded LLM annotations determine the rest.

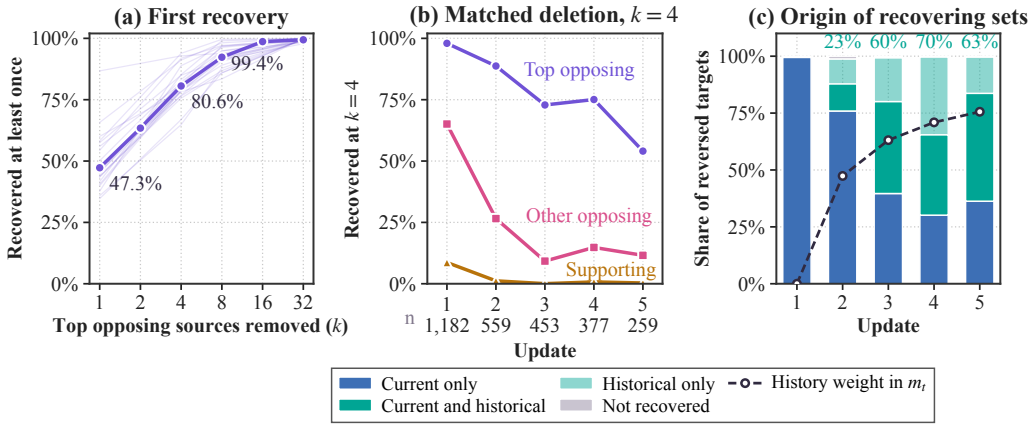


Figure 2: Source-deletion tests. **a**: Cumulative first recovery along the q ranking for all 10,636 reversed targets (thick) and per seed (thin); recovery need not persist after further deletions. **b**: Recovery at $k = 4$ by update, with source count and summed-displacement norm matched ($\pm 10\%$); numbers under the axis are common targets, and random controls average five sets per target. **c**: Reversed targets at each update, grouped by where the members of their recovering set come from; the percentages above the bars are the shares with at least one historical member. The dashed line is the share of first-moment weight carried by past gradients.

Table 1: Relations concentrated among selected sources (%), relative to their candidate pools. Opposite-sign columns report shared operation type; same-sign columns report at least one shared immediate effect. Fractions are computed per target and averaged with equal target weight over the n targets that have selected sources of that kind (19 common seeds). Pools are not filtered by q; no history exists at update 1.

		Opposite sign: same operation			Same sign: shared effect		
Update	Sources	n	Pool	Selected	n	Pool	Selected
1	current	1,437	30.6	**97.2**	750	41.6	**66.5**
2	current	961	35.5	**78.8**	879	36.9	**60.1**
	historical	225	29.3	**97.8**	294	38.0	**73.6**
3	current	931	37.5	**95.9**	811	37.9	**76.6**
	historical	547	31.3	**95.0**	556	37.9	**70.8**
4	current	885	41.5	**86.1**	629	41.0	**84.0**
	historical	695	36.2	**91.4**	862	38.8	**72.9**
5	current	1,120	39.4	**88.9**	1,052	37.7	**70.8**
	historical	745	36.6	**92.7**	820	39.8	**69.5**

Opposite-sign sources often share an operation across different contexts. The task has only four command verbs, so shared operations could reflect the composition of the candidate pool. For each target, we compare the fraction of selected opposite-sign sources that share its operation with the same fraction among all opposite-sign candidates (Table 1). Selected sources share operations more often at every update. Their contexts can still differ: at updates 2–5, 91%–99% of opposite-sign historical pairs share an operation type but only 15%–20% have similar states.

Same-sign sources often share an immediate effect with the target. At the first update, 25.8% of same-sign pairs share the target's operation type, and another 37.6% differ in operation but share an immediate effect, namely `drop`/`put` pairs that free the hand. Shared effects cover 63.4% of raw pairs in total. In the first-update audit, scored by consensus across both input orders, 56.5%–59.2% of deduplicated same-sign pairs are alternatives the policy could have chosen in both states, and 407 of the 533 alternatives play different roles in the task. At that update, together with the negative gradient interaction, these alternatives are consistent with competition between related choices. By update 5, shared effects cover 69.0% of current and 69.3% of historical pairs, almost entirely through

the same operation type, such as `put`/`put` and `take`/`take`. After the first update, members of both signs mostly share the target's kind of operation, and the sign of the gradient interaction separates them. A shared effect need not mean a shared purpose: placing an object in its target container and placing it in a wrong one both free the hand, but only the former completes a goal.

The same target-level comparison for shared effects also shows concentration among selected same-sign sources at every update (Table 1). Both candidate baselines control source composition by count, without matching credit magnitudes, gradient norms, shared token prefixes, or exact observations. They establish concentration, not a causal link between the relations and the gradient interactions.

Appendix E gives one verified case of each relation from the first update (Figure 4); in both, removing the top source alone restores the target's credit direction.

Because historical sources appear in the recovering sets of about half of the reversals after the first update, how much history the optimizer retains deserves its own study, which Section 4 takes up in agent training.

4 From Analysis to Training Intervention: Controlling History Retention

4.1 From Tracing Effects to Training Design

Section 3 shows that past credit, carried by momentum, takes part in present action changes. Whereas the deletion tests examine selected contributions at a fixed update, we now ask how an aggregate retention adjustment changes learning throughout agent training.

Per-action gradient decomposition is useful for explaining updates but too costly for routine training. We instead construct historical and current displacements from Adam's existing moments and current total gradient (Kingma & Ba, 2015). Let t denote an optimizer update, g_t the total-loss gradient after existing clipping, and D_t the diagonal scaling after updating the second moment with this gradient. Computing norms over all participating parameters, we define

$$H_t = -\frac{\eta_t \beta_1}{1-\beta_1^t} D_t m_{t-1}, \qquad C_t = -\frac{\eta_t (1-\beta_1)}{1-\beta_1^t} D_t g_t,$$
$$\rho_t = \min\left(1, \kappa \frac{\|C_t\|_2}{\|H_t\|_2}\right), \quad \widetilde{m}_{t-1} = \rho_t m_{t-1}. \tag{6}$$

For $\|H_t\|_2 = 0$, set $\rho_t = 1$. After gradient clipping and before `optimizer.step()`, we write $\widetilde{m}_{t-1}$ into persistent `exp_avg`, then run AdamW normally (Loshchilov & Hutter, 2019). History changes from H_t to $\rho_t H_t$; the current displacement C_t, scaling D_t, and decoupled weight-decay term remain unchanged in this update.

The modification persists beyond this update: with m_{t-1} denoting history before scaling, $m_t = \beta_1 \rho_t m_{t-1} + (1-\beta_1) g_t$, so past signals acquire every intervening factor ρ_t. In terms of Eq. (3), this weakens all retained contributions uniformly rather than the opposing sources identified in Section 3. We test $\kappa = 2$ (`cap2`). Because the bias correction $1-\beta_1^t$ is left unchanged, attenuation also shortens the step whenever consecutive gradients agree (Appendix F.2).

4.2 Testing the Intervention in Agent Training

We apply the intervention to GiGPO (Feng et al., 2025) and GRPO (Shao et al., 2024) on ALFWorld (Shridhar et al., 2021) and WebShop (Yao et al., 2022), training all parameters of Qwen2.5-1.5B-Instruct and keeping the PPO loss, KL regularization, gradient clipping, and AdamW of the original recipes (Schulman et al., 2017; Loshchilov & Hutter, 2019). For each algorithm–environment pair, `original` runs the unmodified optimizer and `cap2` applies Eq. (6) at two or three learning rates (seed 0, 200 iterations). Table 2 reports per-category success rates from formal evaluation at the final checkpoint; Appendix F gives full training curves and all validation values.

Uniform attenuation does not isolate the opposing sources. Section 3 showed that the sources opposing a given target are a small, identifiable subset of what momentum carries; the remainder

Table 2: Per-category formal evaluation at iteration 200 (seed 0). GiGPO on ALFWorld pools both learning rates (548 episodes); the other three settings use the better learning rate per arm (ALFWorld 274 episodes, WebShop 500 tasks). Bold marks the better arm; ties are unmarked.

		Put	Exam.	Clean	Cool	Heat	2-obj	All
GiGPO	`original`	89.8	93.5	90.5	**90.2**	**96.2**	**85.4**	**90.7**
	`cap2`	**93.2**	93.5	**93.1**	89.1	87.2	82.9	90.1
GRPO	`original`	76.3	71.0	**82.8**	60.9	76.9	**58.5**	71.9
	`cap2`	**83.1**	71.0	77.6	**78.3**	**84.6**	56.1	**75.9**

		Beauty	Electr.	Clothing	Home&G.	Food	All
GiGPO	`original`	57.9	70.6	68.0	72.9	53.8	68.0
	`cap2`	**68.4**	70.6	**78.0**	**78.6**	**76.9**	**77.4**
GRPO	`original`	**68.4**	52.9	**68.0**	70.0	**61.5**	**67.6**
	`cap2`	47.4	52.9	64.3	**72.9**	46.2	64.0

supports the credit direction or acts on other targets. Because `cap2` scales all retained contributions equally, it weakens both subsets in proportion and its net effect depends on the balance between them. The four settings in Table 2 bear this out: `cap2` raises the total in two (GRPO–ALFWorld $+4.0$, GiGPO–WebShop $+9.4$ points), lowers it in one (GRPO–WebShop -3.6), and leaves it essentially unchanged in one (GiGPO–ALFWorld -0.6). Within each setting the per-category pattern is also mixed: in GiGPO–WebShop, `cap2` gains in four of five product categories; in GRPO–WebShop, the loss concentrates in Beauty and Food. The direction thus varies with the algorithm, the task distribution, and the category, as expected when the intervention cannot distinguish opposing from consistent history. Removing only the opposing components identified in Section 3, or renormalizing the attenuated moment to separate direction from step length (Appendix F.2), would test whether selective intervention succeeds where uniform attenuation does not.

5 Related Work

Comparison conditions and feedback for credit assignment. Most step-level credit comes from a comparison: branching rollouts (Kazemnejad et al., 2025; Ji et al., 2026), matching observations that recur across trajectories (Feng et al., 2025; He et al., 2026), or learning step rewards from trajectory preferences (Liu et al., 2026; Xi et al., 2026). We take GiGPO's construction as given and study how its credit acts once it enters training.

Joint learning and the effects of credit. Several lines of work study how credit on one output acts on others (Ren & Sutherland, 2025; Razin et al., 2025; Deng et al., 2025; Sullivan & Koller, 2026). Closest to ours, the Cancellation Hypothesis explains advantage–probability discrepancies through token coupling (Cheng et al., 2026), and Liu et al. (2025) study cross-sample transfer in agents and reduce it with behavioral classification. These works mainly examine the current batch; we also trace history carried by momentum and test located sources against matched controls.

Training data attribution and optimizer state. TracIn and TRAK attribute model changes to training points via gradient products or linearized projections (Pruthi et al., 2020; Park et al., 2023). Trajectory-based methods retrace sample removal through SGD (Hara et al., 2019), including momentum (Chen et al., 2021). Recent work accounts for Adam state within a batch (Ding et al., 2026) or replaces SGD dynamics with AdamW's (Deng et al., 2026). Our analysis expands the actual first moment into per-action records from current and earlier batches and tests deletions within the observed displacement.

Optimizer history under changing learning objectives. Asadi et al. (2023) study state resets after objective changes; Adam-Rel retains moments but resets local bias-correction time (Ellis et al., 2024). In group-based LLM RL, Fontana et al. (2026) examine reward scaling and momentum crossing clipping boundaries during repeated optimization, and Mukherjee et al. (2026) find that AdamW's

momentum and adaptive learning rates matter less in RL than in supervised fine-tuning, with SGD matching AdamW while updating few parameters.

6 Discussion and Conclusion

This paper examined how credit signals, once they enter training, change the probabilities of specific actions. In our GiGPO runs, 37.5% of actions with nonzero credit moved against it. Where count- and norm-matched controls exist, removing the most strongly opposing sources restored the credit direction far more often than the controls, especially from the second update on (Section 3.2). Compared with candidate actions, the selected opposite-sign sources more often shared the target's kind of operation, and the selected same-sign sources one of its immediate effects. This suggests two routes by which credit may reach other actions and indicates which relations to examine when probability changes disagree with credit. Our evidence comes from one controlled environment with five updates per seed, and the audit measures the concentration of these relations rather than their mechanism. Competition between alternatives, matched replacement of historical sources, and norm-rescaled deletion remain to be tested. Within these limits, one finding matters for training: after the first update, about half of all reversals have recovering sets with historical members.

History retention is therefore a design choice alongside current credit. Across four algorithm–environment settings, attenuating all retained momentum had no consistent effect on final success. Unlike switching to SGD (Mukherjee et al., 2026), it keeps Adam's normalization. Whether removing only the opposing sources of Section 3 helps remains untested.

AI use statement

In this work, we used generative AI tools for several tasks that require disclosure. A frozen Qwen3.8-27B-FP8 model (Qwen Team, 2026b) served as a blinded annotator in the semantic audit (Section 3.3); its inputs and disagreement handling are reported in Appendix E and its prompts in Appendix G. LLM coding assistants helped implement and run the experiments and the analysis code under the authors' direction. The manuscript was first written in Chinese and translated into English with LLM assistance. We have not used generative AI tools to generate synthetic data sets or to clean or reformat data sets, and proofs are not applicable to this work. Additionally, we used generative AI tools to help draft, edit, and polish the manuscript and to write figure- and table-generation code. The research questions, hypotheses, experimental designs, and argument of the paper were developed by the authors. We have reviewed all AI-assisted work: the decomposition code is checked against the actual optimizer updates (Appendix C), all reported numbers come from our analysis records and were checked against the text, and the audit prompts and annotation disagreements are reported in full. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

Ethics statement

This work studies training dynamics in a simulated text environment and in ALFWorld and WebShop. It involves no human subjects and no personal data, and the language model used in the audit only annotates simulated interaction records.

Reproducibility statement

Section 2 and Appendix A specify the task generator and its seed, the data splits, rewards, legal actions, GiGPO credit computation, and the full training protocol (Table 4). Appendix B defines the probability measurements and decision rules, Appendix C derives the update decomposition and reports its numerical checks, Appendix D gives the deletion and matched-control protocols with full results, Appendix E gives the audit inputs and handling of disagreements, Appendix F gives the exact optimizer modification, every validation value, and the full evaluation of the retention experiment, and Appendix G reproduces the audit prompts. Upon publication, we will release the code for controlled training, source replay and deletion, the semantic audit, the retention experiment, and table and figure generation, together with the analysis data behind the reported results.

REFERENCES

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

Kavosh Asadi, Rasool Fakoor, and Shoham Sabach. Resetting the optimizer in deep RL: An empirical study. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/e4bf5c3245fd92a4554a16af9803b757-Abstract-Conference.html.

Yuanyuan Chen, Boyang Li, Han Yu, Pengcheng Wu, and Chunyan Miao. HYDRA: Hypergradient data relevance analysis for interpreting deep neural networks. In *AAAI Conference on Artificial Intelligence*, 2021. URL https://arxiv.org/abs/2102.02515.

Tianhao Cheng, Zeyu Huang, Zihan Qiu, Yu Cheng, Edoardo Ponti, Yinghui Xu, Ivan Titov, and Zenglin Xu. The cancellation hypothesis in critic-free RL: From outcome rewards to token credits. arXiv preprint arXiv:2605.08666, 2026. URL https://arxiv.org/abs/2605.08666v1.

Junwei Deng, Pingbang Hu, Suliang Jin, Hao Lu, Jiachen T. Wang, Shichang Zhang, and Jiaqi W. Ma. How faithful is trajectory-based data attribution? error sources, remedies, and practical guidelines. arXiv preprint arXiv:2605.18814, 2026. URL https://arxiv.org/abs/2605.18814.

Wenlong Deng, Yi Ren, Muchen Li, Danica J. Sutherland, Xiaoxiao Li, and Christos Thrampoulidis. On the effect of negative gradient in group relative deep reinforcement optimization. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://openreview.net/forum?id=2K9QsDaqkM.

Meng Ding, Zeqing Zhang, Di Wang, and Lijie Hu. In-run data shapley for Adam optimizer. arXiv preprint arXiv:2602.00329, 2026. URL https://arxiv.org/abs/2602.00329v4.

Benjamin Ellis, Matthew T. Jackson, Andrei Lupu, Alexander D. Goldie, Mattie Fellows, Shimon Whiteson, and Jakob N. Foerster. Adam on local time: Addressing nonstationarity in RL with relative Adam timesteps. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/f2733d3b0dde1d74995f35a9cf442d38-Abstract-Conference.html.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/420c9f777c0b4f78d515e53cf74d58b2-Abstract-Conference.html.

Aleksandar Fontana, Marco Simoni, Giulio Rossolini, Paolo Mori, and Andrea Saracino. On the hidden objective biases of group-based reinforcement learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 109–121, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-short.11. URL https://aclanthology.org/2026.acl-short.11/.

Satoshi Hara, Atsushi Nitanda, and Takanori Maehara. Data cleansing for models trained with SGD. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. URL https://arxiv.org/abs/1906.08473.

Shuo He, Lang Feng, Qi Wei, Xin Cheng, Lei Feng, and Bo An. Hierarchy-of-groups policy optimization for long-horizon agentic tasks. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=T8Dev99qnz.

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.

Yuxiang Ji, Ziyu Ma, Yong Wang, Guanhua Chen, Xiangxiang Chu, and Liaoni Wu. Tree search for LLM agent reinforcement learning. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/8c7304e77c832ddc70075dfee081ca6c-Abstract-Conference.html.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Refining credit assignment in RL training of LLMs. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 29557–29590. PMLR, 2025. URL https://proceedings.mlr.press/v267/kazemnejad25a.html.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

Jiazheng Li, Yawei Wang, Qiaojing Yan, Yijun Tian, Zhichao Xu, Huan Song, Panpan Xu, and Lin Lee Cheong. SALT: Step-level advantage assignment for long-horizon agents via trajectory graph. In *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 4709–4725, Rabat, Morocco, March 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-eacl.247. URL https://aclanthology.org/2026.findings-eacl.247/.

Jingyu Liu, Xiaopeng Wu, Jingquan Peng, Kehan Chen, Chuan Yu, Lizhong Ding, and Yong Liu. Gradient coupling: The hidden barrier to generalization in agentic reinforcement learning. arXiv preprint arXiv:2509.23870, 2025. URL https://arxiv.org/abs/2509.23870v4.

Xiaoqian Liu, Ke Wang, Yuchuan Wu, Fei Huang, Yongbin Li, Jianbin Jiao, and Junge Zhang. Agentic reinforcement learning with implicit step rewards. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ooROvpmxMV.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Xufang Luo, Yuge Zhang, Zhiyuan He, Zilong Wang, Siyun Zhao, Dongsheng Li, Luna K. Qiu, and Yuqing Yang. Agent lightning: Train ANY AI agents with reinforcement learning. arXiv preprint arXiv:2508.03680, 2025. URL https://arxiv.org/abs/2508.03680v1.

Sagnik Mukherjee, Lifan Yuan, Pavan Jayasinha, Dilek Hakkani-Tür, and Hao Peng. Do we need Adam? surprisingly strong and sparse reinforcement learning with SGD in LLMs. In *International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2602.07729v2.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. TRAK: Attributing model behavior at scale. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 27074–27113. PMLR, 2023. URL https://proceedings.mlr.press/v202/park23c.html.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/e6385d39ec9394f2f3a354d9d2b88eec-Abstract.html.

Qwen Team. Qwen3.5-0.8B model card. Hugging Face model card, 2026a. URL https://huggingface.co/Qwen/Qwen3.5-0.8B. Accessed 2026-09-25.

Qwen Team. Qwen3.8-27B-FP8 model card. Hugging Face model card, 2026b. URL https://huggingface.co/Qwen/Qwen3.8-27B-FP8. Accessed 2026-09-25.

Noam Razin, Sadhika Malladi, Adithya Bhaskar, Danqi Chen, Sanjeev Arora, and Boris Hanin. Unintentional unalignment: Likelihood displacement in direct preference optimization. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/3df38ca67befaed9c03b95ffee07d9f8-Abstract-Conference.html.

Yi Ren and Danica J. Sutherland. Learning dynamics of LLM finetuning. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/afe1aa79e5eea7955f553c61a307273e-Abstract-Conference.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017. URL https://arxiv.org/abs/1707.06347v2.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. arXiv preprint arXiv:2402.03300, 2024. URL https://arxiv.org/abs/2402.03300v3.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.03768.

Michael Sullivan and Alexander Koller. GRPO is secretly a process reward model. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2509.21154v4.

Hanlin Wang, Chak Tou Leong, Jiashuo Wang, Jian Wang, and Wenjie Li. SPA-RL: Reinforcing LLM agents via stepwise progress attribution. arXiv preprint arXiv:2505.20732, 2025a. URL https://arxiv.org/abs/2505.20732.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Fei-Fei Li, Lijuan Wang, Yejin Choi, and Manling Li. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning. arXiv preprint arXiv:2504.20073, 2025b. URL https://arxiv.org/abs/2504.20073v2.

Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, Hyokun Yun, and Lihong Li. WebAgent-R1: Training web agents via end-to-end multi-turn reinforcement learning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 7909–7928, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.401. URL https://aclanthology.org/2025.emnlp-main.401/.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992. doi: 10.1007/BF00992696. URL https://link.springer.com/article/10.1007/BF00992696.

Zhiheng Xi, Chenyang Liao, Guanyu Li, Yajie Yang, Wenxiang Chen, Zhihao Zhang, Binghai Wang, Senjie Jin, Yuhao Zhou, Jian Guan, Wei Wu, Tao Ji, Tao Gui, Qi Zhang, and Xuanjing Huang. AgentPRM: Process reward models for LLM agents via step-wise promise and progress. In *Proceedings of the ACM Web Conference 2026*, 2026. doi: 10.1145/3774904.3792551. URL https://doi.org/10.1145/3774904.3792551.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. ArCHer: Training language model agents via hierarchical multi-turn RL. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62178–62209. PMLR, 2024. URL https://proceedings.mlr.press/v235/zhou24t.html.

A Controlled Environment and Training Protocol

This section provides the complete task rules, input format, credit computation, and update protocol for the controlled training runs. The environment state determines which commands can be executed and their immediate effects. Interpretations such as preparation, freeing the hand, completing a goal, or undoing progress can therefore be checked against explicit state transitions rather than inferred model intentions.

A.1 States, Actions, and Task Completion

The environment contains six objects: `apple`, `book`, `cup`, `key`, `pen`, and `bottle`; three containers: `box`, `basket`, and `cabinet`; a table; and one hand. Each task specifies three distinct goal objects and their respective target containers, using at least two distinct target containers. Each object is in exactly one location: on the table, in the hand, or inside a container. The hand can hold at most one object; all containers and the table have unlimited capacity in this experiment. Containers can be open or closed, but there is no command to close them.

An object's goal is complete when it is in its designated container, regardless of whether that container is open. Success requires all three goals to hold simultaneously, rather than to have been completed at some point. Some goals may already be complete in the initial state; taking one of these objects out undoes that progress. Apart from object locations and container states, there are no hidden stochastic transitions.

Table 3: Complete action rules. o denotes an object and c a container. Physical executability does not depend on goal completion; the same executable command can advance a task or undo progress.

Command	Preconditions	State change
`open` c	c is closed; the hand need not be empty	c opens; object locations do not change
`take` o	The hand is empty; o is on the table or in an open container	o moves to the hand; if it was in its target container, that goal is undone
`drop` o	The hand holds o	o moves to the table, leaving the hand empty
`put` o `in` c	The hand holds o and c is open	o moves into c, leaving the hand empty; if c is its target container, that goal is completed

Each trajectory allows at most 14 commands. Each step first applies the state transition and then checks whether all three goals are complete; the trajectory ends on success or when the action budget is exhausted. Let s_{n+1} be the state after command n. The immediate reward is

$$r_n = \mathbf{1}\{s_{n+1} \text{ satisfies all goals}\} - 0.01. \tag{7}$$

The successful step therefore receives 0.99 and is not exempt from the step cost. A successful trajectory of length T has total return $1 - 0.01T$; a failed trajectory has total return $-0.01T$. The action mask permits only the executable commands in Table 3. An illegal command raises an error in the environment implementation; it is not replaced by a default action or assigned an additional reward.

A.2 Task Generation, Splits, and the Shared Task Queue

The task pool has nine strata formed by crossing two factors: the number of initially completed goals (0, 1, or 2), and the initial hand state (empty, holding an unfinished goal object, or holding an irrelevant object). Each stratum contains 144 distinct tasks, giving 1,296 in total. The generator first samples three goal objects and their target containers, excluding configurations in which all three goals use the same container. It then selects the initially completed goals required by the stratum and places the remaining goal objects outside their target containers. Locations for objects not held in the hand are sampled from the table and the three containers, and container open/closed states are generated independently. For strata that require a held object, the generator then moves an appropriate object to the hand; a held goal object is selected only from unfinished goals.

Tasks are deduplicated by the joint content of their goals, initial physical state, and action limit. An environment planner checks that each task has a completion sequence of at most 14 steps and excludes tasks requiring no action or only one step. Neither the plan nor its length is provided to

the policy. The task-generation seed is 20260923. Within each stratum, tasks are sorted by a fixed content hash and divided into 8 calibration, 16 test, and 120 training tasks, yielding 72 calibration, 144 test, and 1,080 training tasks. The complete task contents do not overlap across these splits, but the object vocabulary, action rules, and generation strata are shared. This is not an out-of-domain split by vocabulary or task type.

Each split is shuffled in a fixed random order, and three distinct tasks are drawn from the training pool for each update; tasks may recur across updates. Training seeds share the initial model, task split, and predetermined task queue. Their sampling streams are distinguished by training seed, update, task slot, and resampling attempt. The accepted tasks can differ under the zero-variance resampling rule below, so the final training batches across seeds are not treated as identical experiences.

A.3 Policy Inputs, Command Generation, and Action Probabilities

The policy uses the model's original chat template. A system message specifies the environment rules, and a user message supplies the task, all object locations, container states and capacities, steps used, the actual action history, and the currently executable commands. The system message is reproduced below. Thinking output is disabled, and the model generates only one command.

Policy System Prompt

```
You control a deterministic object-placement environment.
Complete ALL placements specified in the task at the same time.
There are six objects (apple, book, cup, key, pen, bottle), three
containers (box, basket, cabinet),
a table, and one hand. All object locations and container states are
visible.
You can hold at most one object. The table has no capacity limit.
take OBJECT: take an accessible object into your empty hand. An object
in a
closed container is inaccessible until that container is opened.
drop OBJECT: put the object in your hand on the table.
open CONTAINER: open a closed container; your hand need not be empty.
put OBJECT in CONTAINER: put the object in your hand into an open
container
with enough free capacity. The observation states each container's
capacity.
Choose exactly one of the available commands. Output only that
command.
```

The following user message precedes the target action in Case E.7. Line breaks are adjusted only for presentation; the factual fields and command list retain the actual input format. The example also shows why the policy's input history and the observation used for GiGPO matching are different objects.

Example Policy Input

```
# Task
put book in cabinet . put cup in box . put pen in basket .

# Current observation
apple at cabinet . book at box . cup at box . key at table . pen at
basket . bottle at
basket . box is closed . basket is closed . cabinet is open . box
capacity unlimited .
basket capacity unlimited . cabinet capacity unlimited . hand capacity
1 .

# Action budget
1 used out of 14.
```

```
# Previous actions
1. put apple in cabinet

# Available commands
- open basket
- open box
- take apple
- take key
```

All physical state is visible. The model receives neither the planner's completion path nor the shortest completion length, action-quality labels, or future returns. The history lists preceding commands, while the current observation describes the resulting physical state. Inputs are limited to 2,048 tokens; exceeding the limit raises an error rather than truncating the history.

Each currently legal command is encoded with one end token appended to its response, forming a command prefix tree. At response token k, normalization is restricted to the set $\mathcal{V}_k$ of legal successors compatible with the generated prefix. If the original model's logit is z_v, the token probability used in this experiment is

$$\pi_\theta(v \mid h, a_{<k}, \mathcal{V}_k) = \frac{\exp z_v}{\sum_{w \in \mathcal{V}_k} \exp z_w}, \quad v \in \mathcal{V}_k. \tag{8}$$

Training samples directly from this distribution at temperature 1, without top-k or top-p filtering. Evaluation chooses the most probable legal successor at each token. A token with only one legal successor has probability 1. The probability of a complete command is the product of its response-token probabilities, including the end token; it is not a softmax over separately scored candidate commands. Sampling, loss computation, and measurements before and after an update use the same mask and response definition.

A.4 GiGPO Credit Computation

Eight trajectories are sampled per task to form a G8. Let trajectory ξ have length T_ξ and total environment return $G_\xi = \sum_n r_{\xi,n}$. We use `mean_norm` from the original GiGPO implementation, with the trajectory mean computed across action records: a trajectory's total return appears once for each of its action records. The G8 reference mean and episode term are therefore

$$\overline{G}_g = \frac{\sum_{\xi \in g} T_\xi G_\xi}{\sum_{\xi \in g} T_\xi}, \qquad E_\xi = G_\xi - \overline{G}_g. \tag{9}$$

This preserves the original reduction over action records, rather than averaging the eight trajectories equally, and does not divide by the standard deviation.

Action i has discounted continuation return $R_i^\gamma = \sum_{n=i}^{T_\xi - 1} \gamma^{n-i} r_{\xi,n}$, with $\gamma = 0.95$. Its reference set $\mathcal{R}_i$ contains all visits in the same G8 with exactly the same pre-action observation string, including itself and repeated visits within its own trajectory. Records are not pooled across tasks, and no approximate text matching is used. The observation string contains object locations, container states, and capacities, but not previous commands or steps used. The step term and total credit are

$$S_i = R_i^\gamma - \frac{1}{|\mathcal{R}_i|} \sum_{j \in \mathcal{R}_i} R_j^\gamma, \qquad A_i = E_{\xi(i)} + S_i. \tag{10}$$

The step-term weight is 1, and this term is also centered without standard-deviation scaling. It is zero when there is only one visit or all reference continuation returns are equal. Matching observations need not have matching histories or remaining budgets. Credit is fixed after the batch has been collected; semantic auditing does not enter its computation.

A.5 Batch Acceptance, Parameter Updates, and Evaluation

Each update accepts three G8s, totaling 24 trajectories. If a G8's eight raw total returns satisfy $\max G_\xi - \min G_\xi \le 10^{-8}$, the entire group is excluded from the loss and resampled on the same

Table 4: Main settings for controlled training. Parameters and optimizer state are carried across updates; this appendix gives the full protocol.

Item	Setting
Model and parameters	Qwen3.5-0.8B; LoRA rank/alpha both 8; no additional SFT
Batch sampling	Three tasks, each G8 (eight trajectories per task)
Credit	GiGPO, $\gamma = 0.95$, step weight 1; group mean centering without standard-deviation scaling
Optimization	Adam, $\eta = 3 \times 10^{-4}$, $(\beta_1, \beta_2) = (0.9, 0.999)$; FP32
Fixed task split	1,080 training, 72 calibration, 144 test
Analysis window	21 seeds $\times$ 5 updates, plus the first update of one more seed: 106 updates, 28,363 action records (about 270 per update)

task, for at most 32 groups. The next task is then taken from a predetermined replacement queue drawn from the original training pool. This queue excludes tasks in the predetermined schedule and assigns replacements to task slots in a fixed order. Training pauses if the queue is exhausted, rather than executing an empty update. This selection uses only environment returns, not labels or source analysis. Equal success rates do not imply equal returns: successful trajectories of different lengths can still provide return differences.

Parameters remain fixed during sampling. All accepted actions share the full-batch denominator Z_t, the number of valid response tokens including end tokens; each action's total credit is broadcast unchanged to its response tokens. Each compute process accumulates loss gradients under this global normalization. These gradients are summed on the CPU in FP64 and converted back to FP32 before the main process executes a single `torch.optim.Adam` update. There are no multiple PPO epochs, gradient clipping, or additional losses; parameters and both moment states are retained across updates. Individual action gradients are not computed during training, but replayed later at the saved original parameter states. Table 5 lists the full configuration.

Table 5: Controlled training and evaluation configuration. The model backbone is frozen; only LoRA parameters in the language layers' linear modules are updated.

Item	Configuration
Model	Qwen3.5-0.8B; revision `2fc06364715b967f1860aea9cf38778875588b17`
LoRA	rank=8, alpha=8, dropout=0, no bias; initialization seed 20260908
Adapted modules	All linear modules in the language-model layers, including attention and feed-forward layers; embeddings, vocabulary projection, and visual components are frozen
Extra initialization	No SFT or demonstration training
Input and generation	At most 2,048 tokens; original chat template; thinking output disabled; one legal command followed by an end token
Trajectories and batches	At most 14 actions; 3 tasks$\times$G8; one Adam update per batch
GiGPO	$\gamma = 0.95$, step-term weight 1, mean centering, exact observation matching
Adam	Learning rate 3×10^{-4}, betas $(0.9, 0.999)$, eps 10^{-8}
Additional terms	No weight decay, KL, entropy bonus, or gradient clipping
Precision	FP32 model and optimizer; gradients summed across processes in FP64, then cast to FP32
Evaluation	After every update, the same 144 tasks; one token-wise greedy trajectory per task

Evaluation records success, total return, and trajectory length, including failed trajectories in the mean length. Random-number states are saved and restored around evaluation so that it does not consume the subsequent training sampling stream. Seeds share the initial LoRA parameters rather than starting from separately pretrained models; their paths diverge through sampling and subsequent updates. Appendix B gives the analysis windows and observations for each update.

B Analysis Scope and Action Probability Changes

This section specifies the records used for probability measurements, deletion experiments, and experience audits, states the statistical units and weightings used throughout the paper, and reports raw counts and test performance by seed and update.

B.1 Training Windows and Analysis Subsets

Probability measurements and deletion experiments cover the first five updates of seeds 1–9 and 11–22, plus the first update of seed 10, for 106 updates in total. Learning curves use the 21 seeds with five consecutive updates. Seed 10 has a complete record only for its first update; no values are imputed for later positions. Table 6 lists each analysis subset and its use. Each update contains 24 trajectories that actually entered the optimizer, giving 2,544 trajectories and 28,363 action records across all windows. Resampled trajectories that were not accepted are excluded from these counts.

Table 6: Analysis samples and their uses. Historical sources may precede the target update; update ranges here refer to the targets being explained. Rows are related analysis subsets, not disjoint groups.

Analysis	Seeds and target updates	Statistical units and counts
Probability measurements	Update 1 of 22 seeds; updates 2–5 of 21 seeds	106 updates, 28,363 actions
Source deletion	Same range, restricted to observed reversals	10,636 targets; fixed-k controls use the common matched subset
Learning context	Seeds 1–9 and 11–22, updates 1–5	105 checkpoints, each evaluated on 144 tasks
First-update relation audit	Seeds 1–20, update 1	1,711 original opposite-sign relations; 944 deduplicated same-sign inputs
Later-update relation audit	Seeds 1–9 and 11–22, updates 2–5	8,216 reversed targets; 28,318 relations in recovery sets
Cross-stage relation comparison	Seeds 1–9 and 11–20, updates 1–5	19 common seeds; original source–target occurrences are counted

The cross-update comparisons in the main text use the intersection of seeds in the first-update and later-update audits, so that changes in seed composition are not mistaken for changes across training stages. Deduplicating same-sign inputs from the first update allows the same annotation result to be reused. They are mapped back to the original relation records before comparison with later updates. Appendix E gives the denominators and annotation summaries.

B.2 Statistical Units, Weighting, and Scope

The following conventions apply to every appendix; later sections refer back to them instead of restating them.

- **Units.** Probability measurements and deletion experiments count action records (a deletion target is a reversed record). Relation statistics count source–target pairs. The first-update same-sign audit and the order checks count deduplicated annotation inputs. Rates in different units are not added together, and none serves as the denominator of another. In particular, relation rates are not fractions of all reversals explained, not evidence that each member of a recovery set is necessary, and not shares of the probability change.
- **Weighting.** A pooled rate weights records (or targets) equally; an equal-target rate first computes a per-target fraction and then averages targets; an equal-seed rate first averages within each seed and then weights seeds equally. These answer different questions and generally differ. Unless stated otherwise, the main text uses pooled rates for records and targets and equal-target rates for relation concentration; equal-seed values are reported as checks.

- **Dependence.** Records from the same batch, pairs sharing a target, and the five random control sets of one target are not independent replicates. Bootstrap intervals for the controlled runs resample seeds.
- **Scope of relation statistics.** Audited pairs are selected through reversal events and first-recovery sets, not drawn at random. Comparisons with candidate pools show concentration of a relation among selected sources; they do not show that the relation causes the gradient interaction.

B.3 Forward-Pass Measurements and Direction Classification

For training action i, we compute its conditional log-probability at the pre-update parameters θ_{t-1} and post-update parameters θ_t, using its actual input tokens, complete observed response, and legal successor set at each token. Both computations score the fixed response token by token; they neither regenerate an action nor change its history. Scoring uses the same prefix length and mask as sampling, so that input-layout changes are not mixed with parameter-update effects.

The complete action's $\ell_{t,i}(\theta)$ is the sum of its valid response-token log-probabilities, and its probability is $\exp \ell_{t,i}(\theta)$. Direction is classified using $\Delta\ell$, rather than rounded displayed probabilities. Credit is treated as zero when $|A_{t,i}| \le 10^{-12}$. For the remaining records, strictly positive, negative, and zero values of $r_{t,i} = \operatorname{sgn}(A_{t,i})\Delta\ell_{t,i}$ are classified as aligned, reversed, and unchanged, respectively, with no additional dead zone for probability changes. Very small nonzero changes therefore retain their measured direction; "unchanged" does not include all practically negligible changes.

Occurrences of the same input and action in different trajectories remain distinct records because each occurrence's credit actually enters the loss. The proportions reported here describe these training records; the records are not treated as independent experimental replicates. Below, we also merge identical loss terms to check whether separately counting repeated records accounts for most events. Source deletion and relation auditing still use the original records and frozen recovery sets; their analysis targets are not replaced by the classifications obtained after merging.

For seed e at update t, let $B^+_{e,t}$ denote the set of nonzero-credit records. The reversal rate for that update is

$$p_{e,t} = \frac{\sum_{i \in B^+_{e,t}} \mathbf{1}\{r_{e,t,i} < 0\}}{|B^+_{e,t}|}. \tag{11}$$

Figure 1b computes $p_{e,t}$ first, then averages it equally across seeds with records at that update. The overall proportion in the main text instead divides the total number of reversed actions by the total number of nonzero-credit actions. The former weights seeds equally; the latter weights action records equally, so the two quantities are not interchangeable. Zero-credit records are kept as a separate category; none occur in this analysis window.

B.4 Observations by Update and Seed

Of the 28,363 actions, 17,706 change in the direction indicated by their own credit, 10,636 change in the opposite direction, and 21 remain exactly unchanged. The pooled reversal rate is 37.50%. Table 7 pools these counts by update, and Table 8 gives the numerator and denominator for every seed and update, allowing both the overall rate and the curve averaged equally across seeds to be checked. These changes are not judged beneficial or harmful using semantic labels.

Table 7: Observed probability changes by update. Rates pool the action counts in each row; Figure 1b weights seeds equally.

Update	Seeds	Actions	Aligned	Reversed	Unchanged	Reversal rate
1	22	5,987	3,567	2,420	0	40.42%
2	21	6,481	4,385	2,096	0	32.34%
3	21	5,712	3,712	2,000	0	35.01%
4	21	5,120	3,076	2,044	0	39.92%
5	21	5,063	2,966	2,076	21	41.00%

Table 8: Reversed records / nonzero-credit actions by seed and update. Only the first update of seed 10 is included; all other seeds contribute five updates.

Seed	Update 1	Update 2	Update 3	Update 4	Update 5
1	98/264	127/283	102/298	96/222	124/288
2	101/271	109/305	100/302	108/325	96/298
3	99/291	132/323	107/301	121/281	103/280
4	120/282	78/328	100/293	93/225	82/253
5	110/277	90/311	82/237	67/214	86/182
6	104/271	110/279	102/243	77/200	98/218
7	128/273	86/276	94/208	71/198	71/196
8	108/264	94/304	98/298	55/182	56/169
9	109/274	121/282	79/218	77/206	94/172
10	98/277	First update only			
11	95/238	111/308	72/311	121/301	89/226
12	110/260	124/319	100/234	86/224	110/229
13	124/269	76/306	102/309	119/269	105/258
14	118/271	69/331	104/254	156/312	138/308
15	121/276	132/303	85/243	93/230	65/208
16	119/261	78/301	77/241	61/176	84/235
17	99/265	112/313	75/311	129/309	147/290
18	117/273	89/327	108/329	119/289	118/261
19	120/296	90/315	83/234	61/206	81/201
20	99/260	105/314	110/303	104/254	127/276
21	115/315	108/328	105/287	155/301	108/261
22	108/259	55/325	115/258	75/196	94/254

B.5 Merging Identical Loss Terms

Repeated records can receive different credit while corresponding to the same probability. Within each seed and update, we therefore form a group G from records with identical input-token sequences, complete response-token sequences, and legal successor sets at every token, and set $A_G = \sum_{i \in G} A_i$. The loss retains the original batch response-token count Z_t, without renormalizing by the deduplicated token count. This combines only the coefficients of identical loss terms; it does not change the training objective, actual update, or saved probabilities. Records from different updates are not merged because their parameter states and gradients differ.

We verify programmatically that pre-update, post-update, and difference log-probabilities are identical within every group, then reclassify the term's direction using the sign of its combined credit. There are 3,236 groups containing repeated occurrences, totaling 11,018 records; 1,536 groups contain both positive and negative credit. Table 9 shows a reversal rate of 36.92% after merging, close to the original record-level rate of 37.50%. The corresponding averages with equal seed weights are 36.86% and 37.44%. Separately counting repeated loss terms therefore cannot by itself explain the reversals in this window.

Table 9: Reversal rates before and after merging identical loss terms. Denominators are nonzero-credit records or merged terms in the corresponding columns; neither representation has zero-credit items in this window.

	Original action records			Merged loss terms		
Update	Total	Reversed	Rate	Total	Reversed	Rate
1	5,987	2,420	40.42%	5,407	2,142	39.62%
2	6,481	2,096	32.34%	4,284	1,422	33.19%
3	5,712	2,000	35.01%	4,026	1,337	33.21%
4	5,120	2,044	39.92%	3,642	1,439	39.51%
5	5,063	2,076	41.00%	3,222	1,258	39.04%
Total	28,363	10,636	37.50%	20,581	7,598	36.92%

Merging also changes the reference used to classify some records. Counting original records, 1,599 of the 10,636 originally reversed records become aligned with the combined credit, while 9,037 remain reversed. Of the 17,706 originally aligned records, 1,655 become reversed and 16,051 remain aligned. The 21 originally unchanged records remain unchanged. Only the credit sign used for comparison changes, not the action probability; the 1,599 records cannot be interpreted as a further recovery produced by training.

We also inspect the 10,573 frozen first-recovery sets. Of these, 562 contain other records with exactly the same current loss term as the target (5.32%), and 252 consist entirely of such records (2.38%). Directly repeated sources are thus one case among the localized sources, rather than an explanation for all of them. We have not rerun deletion after excluding repeated sources, so this count does not establish an independent cross-context effect for every source in the remaining sets.

B.6 Test Performance of the Same Checkpoints

Every evaluation uses the same 144 tasks and generates complete trajectories with the token-wise greedy policy described in Appendix A. Table 10 lists the number of tasks solved by each checkpoint, starting after the first update; this first column is not pre-training performance. Across the 21 seeds trained for five consecutive updates, mean success rises from 31.45% at update 1 to 93.95% at update 5, with fluctuations at intermediate updates and within individual seeds.

Table 10: Number of the fixed 144 tasks solved by each post-update checkpoint. Dividing an entry by 144 gives that checkpoint's success rate.

Seed	Update 1	Update 2	Update 3	Update 4	Update 5
1	72	50	84	106	138
2	4	19	29	62	123
3	34	44	92	121	101
4	22	68	127	119	144
5	68	96	113	137	138
6	66	138	141	142	144
7	69	112	129	139	142
8	32	88	141	143	144
9	120	131	142	144	144
10	0		First update only		
11	13	25	102	124	143
12	43	123	130	133	136
13	35	58	108	121	127
14	13	61	84	103	126
15	89	134	136	142	144
16	58	135	138	134	144
17	49	42	61	85	126
18	37	9	91	125	140
19	38	137	142	142	144
20	45	73	94	133	140
21	12	31	58	74	125
22	32	129	142	133	128

These tests provide the learning context for the probability observations. Reversal events concern training actions scored at fixed inputs, whereas success rates come from full trajectories on independent tasks. They are different outcomes: a reversal for one action does not by itself determine whether a test task succeeds. Across the 84 seed–update pairs at updates 2–5, a higher reversal rate goes with a smaller gain in solved test tasks at the same update (Pearson $r = -0.37$ between the reversal rate in Table 8 and the change in Table 10). Controlling for the update and the previous checkpoint's success, ten more percentage points of reversals correspond to about 6.0 fewer solved tasks, but the seed-bootstrap 95% interval (-13.7 to 2.2) includes zero, so we do not claim that reversals slow learning. Table 11 reports mean return and mean trajectory length per update; all executed steps in failed trajectories are included in the length.

Table 11: Evaluation summary for the 21 seeds with consecutive training updates. Results are averaged over each seed's 144 tasks, then equally across seeds.

Update	Seeds	Success (%)	Mean return	Mean length
1	21	31.45	0.1980	11.65
2	21	56.32	0.4617	10.15
3	21	75.53	0.6669	8.84
4	21	84.72	0.7656	8.16
5	21	93.95	0.8657	7.38

C Action-Source Decomposition and Implementation Checks

The decomposition concerns the actual parameter displacement in the current update, with each source corresponding to an action record that entered training. This section details the Adam expansion, replay at the original parameter states, and numerical checks that connect each source vector to the quantities used during training.

C.1 Expanding the Actual Adam Displacement into Action Contributions

θ contains only LoRA parameters that require gradients; the frozen model backbone is excluded from source vectors. The base action gradient $b_{s,j}$ points locally toward increasing that action's log-probability, while the optimizer receives the loss gradient $g_s = -Z_s^{-1} \sum_j A_{s,j} b_{s,j}$. A historical action's $b_{s,j}$ is evaluated at its original parameter state θ_{s-1}, and its credit $A_{s,j}$ and response-token denominator Z_s retain their original batch values.

For completeness, the action and loss gradients are

$$\begin{aligned} b_{s,j} &= \nabla_\theta \log \pi_\theta(a_{s,j} \mid h_{s,j})|_{\theta=\theta_{s-1}}, \\ g_s &= -\frac{1}{Z_s} \sum_{j \in B_s} A_{s,j} b_{s,j}. \end{aligned} \tag{12}$$

The actual Adam recurrence, with bias correction, is

$$\begin{aligned} &m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \qquad v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^{\odot 2}, \\ &D_t = \operatorname{diag}\left(\frac{1}{\sqrt{v_t/(1-\beta_2^t)} + \epsilon} \right), \qquad \Delta\theta_t = -\frac{\eta_t}{1-\beta_1^t} D_t m_t. \end{aligned} \tag{13}$$

The initial first moment is zero. Expanding the recurrence gives

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t = (1-\beta_1) \sum_{s=1}^{t} \beta_1^{t-s} g_s, \\ \Delta\theta_t &= -\frac{\eta_t}{1-\beta_1^t} D_t m_t \\ &= \sum_{s=1}^{t} \sum_{j \in B_s} \frac{\eta_t (1-\beta_1) \beta_1^{t-s}}{(1-\beta_1^t) Z_s} A_{s,j} D_t b_{s,j}. \end{aligned} \tag{14}$$

The second-moment scaling uses the actual v_t and standard bias correction: $D_t = \operatorname{diag}((\sqrt{v_t/(1-\beta_2^t)} + \epsilon)^{-1})$. Squaring, square roots, and division are coordinatewise. All sources use the current D_t and learning rate, while the historical gradients themselves are not reevaluated. Terms with $s < t$ sum to the historical contribution; terms with $s = t$ sum to the current contribution.

There are no additional losses, gradient clipping, or weight decay in this experiment, so the action losses account for the learning signal passed to Adam. Equation (14) expands the displacement while holding the actual second moment fixed. It neither attributes the formation of that second moment to individual actions nor passes each source separately through a new Adam optimizer. Deleting a source retains this actual scaling; Appendix D describes the resulting finite parameter intervention.

C.2 Recovering and Aligning Sources at Their Original Parameter States

Action gradients are recovered from the saved model parameters, input tokens, responses, and legal successor sets. We differentiate each complete action log-probability independently, without storing a separate parameter gradient for every token. The parameter list follows the model's fixed traversal order of parameters with `requires_grad=True`. Each tensor is flattened according to its stored shape and concatenated into one vector. The same layout is used for parameters, first and second moments, and all action gradients. Records are linked by seed, update, trajectory, and action position, rather than by command name alone.

Computation: from training records to sources for update t.

1. Load the actual pre- and post-update parameters, Adam step count, and post-update second moment for update t to construct D_t.
2. For each $s \leq t$, restore θ_{s-1} and compute each action's $b_{s,j}$ using its original input, response, and mask.
3. Combine $b_{s,j}$ with the record's original $A_{s,j}$, Z_s, and β_1^{t-s} to obtain $U_{s,j}^t$.
4. Sum by source identity and check all sources and historical sources separately; compute q for each current target, exclude its own record, and rank the remaining sources.

The training model and saved base gradients use FP32. Analysis coefficients, Adam scaling, source inner products, and error norms are computed in double precision; interventions are cast back to FP32 when written to the model. During actual training, loss gradients are summed across processes in FP64 before conversion to FP32. Finite-precision reductions for individual-action and whole-batch differentiation need not be bitwise equal. We therefore report numerical errors below rather than treating the algebraic identity as automatic evidence of numerical agreement.

C.3 Own Contributions, Cross-Action Effects, and Finite Changes

For a target with $A_{t,i} \neq 0$ and $\|b_{t,i}\|_2 > 0$, the projection of its own contribution onto the unit direction indicated by its credit is

$$q^t_{i\leftarrow(t,i)} = \frac{\eta_t(1-\beta_1)|A_{t,i}|}{(1-\beta_1^t)Z_t} \frac{b_{t,i}^\top D_t b_{t,i}}{\|b_{t,i}\|_2} > 0. \tag{15}$$

The sign follows from the positive diagonal entries of D_t; cross-inner products with other actions have no such guarantee. Normalizing the target gradient leaves source rankings for that target unchanged. Source credit magnitudes, gradient norms, historical decay, and actual scaling are all retained. Other occurrences of the same input or command also retain their actual training weights; only the target's own current record is excluded.

q is a local projection of a displacement onto the direction indicated by the target's credit. The action log-probability is nonlinear in the parameters, so we compute changes after finite deletion by a forward pass, rather than interpreting q as an exact share of the probability change. The local calculation ranks candidate sources; the experiment in Appendix D checks the selected set's effect under a finite change.

C.4 Displacement Reconstruction and Residuals

For a reconstructed vector x and reference vector y, we use

$$e_{\text{abs}} = \|x-y\|_2, \qquad e_{\text{rel}} = \frac{\|x-y\|_2}{\|y\|_2} \quad (\|y\|_2 > 0). \tag{16}$$

The first check reconstructs the current Adam displacement from the saved actual total loss gradient and pre-update first moment, and compares it with $\theta_t - \theta_{t-1}$. The second compares the sum of all action sources with the same actual displacement. The third compares the sum of historical action sources with the historical displacement obtained from the saved first moment. These checks test the optimizer expression, action expansion, and historical reconstruction, respectively; they are not interchangeable. Table 12 summarizes them over all updates.

The first update has no history. In all 22 first updates, both the historical reference and reconstruction are zero vectors, with zero absolute error and undefined relative error. The remaining 84 updates

Table 12: Relative L_2 errors from the 106 source-replay records. Zero history has no relative-error denominator and is treated separately as a zero vector.

Check	Updates	Median	Maximum
Adam from actual gradient vs. displacement	106	1.29×10^{-6}	1.88×10^{-6}
All action sources vs. displacement	106	3.15×10^{-6}	2.17×10^{-4}
Historical sources vs. saved history term	84	2.33×10^{-6}	1.58×10^{-5}

Table 13: Maximum absolute and relative errors by update. The two maxima are computed separately and need not come from the same record.

Update	Count	Adam (actual total gradient)		All action sources	
		Absolute	Relative	Absolute	Relative
1	22	3.99×10^{-8}	7.47×10^{-8}	1.16×10^{-4}	2.17×10^{-4}
2	21	5.26×10^{-7}	1.11×10^{-6}	8.02×10^{-6}	1.63×10^{-5}
3	21	5.27×10^{-7}	1.35×10^{-6}	1.67×10^{-6}	4.03×10^{-6}
4	21	5.28×10^{-7}	1.62×10^{-6}	1.17×10^{-6}	3.38×10^{-6}
5	21	5.30×10^{-7}	1.88×10^{-6}	9.92×10^{-7}	3.13×10^{-6}

enter the relative-error statistics for nonzero history. Table 13 also reports errors by update so that the overall median does not obscure larger deviations in individual updates. The maximum relative error for reconstruction from all sources is 2.17×10^{-4}, at update 1; Appendix C.5 explains why the first update is most sensitive. This differs from the Adam check using the actual total gradient; neither is presented as bitwise exact reconstruction.

Let the finite-precision residual be $\varepsilon_t = (\theta_t - \theta_{t-1}) - \sum_{s,j} U^t_{s,j}$. Every deletion subtracts the selected sources from the actual θ_t, so the same ε_t remains in all interventions. It is neither assigned to a credit category nor used for ranking. We report the measured reconstruction errors without attributing the unreconstructed part to any action.

C.5 Adam in the Analysis Window

The first update. With $m_0 = v_0 = 0$, bias correction gives $m_1/(1-\beta_1) = g_1$ and $v_1/(1-\beta_2) = g_1^{\odot 2}$, so

$$\Delta\theta_1 = -\eta_1 \frac{g_1}{|g_1| + \epsilon} \approx -\eta_1 \operatorname{sgn}(g_1) \tag{17}$$

on coordinates with $|g_1| \gg \epsilon$. Each such coordinate moves by about η_1 regardless of gradient magnitude, and $D_1 = \operatorname{diag}((|g_1| + \epsilon)^{-1})$ divides every source by the magnitude of the summed batch gradient. Where contributions nearly cancel, $|g_1|$ is small, a single source's $U^1_{1,j}$ can exceed the net displacement on that coordinate, and removing it can flip the coordinate's direction. This mechanism is consistent with three features of the first update, although we have not tested it directly: randomly sampled other opposing sets recover 65.1% of common targets at $k = 4$ (Table 19); removed contributions can be many times the update norm (Figure 4); and the relative error of reconstruction from all sources is largest at update 1 (Table 13). For the last feature, the base gradients are stored in FP32 and computed action by action, whereas the actual update used the batch gradient; the small differences between the two reductions (Appendix C) are divided by $|g_1|$ and are therefore amplified most on nearly cancelling coordinates. A direct test would check whether the displacement removed by recovering deletions concentrates on coordinates with small $|g_1|$. From update 2 onward, v_t averages squared gradients from earlier batches, D_t no longer scales each coordinate by the current gradient alone, and this amplification weakens. For this reason the main text reports updates 2–5 separately from update 1.

Weight carried by past gradients. In Eq. (14), batch s enters m_t with coefficient $(1-\beta_1)\beta_1^{t-s}$. These coefficients sum to $1-\beta_1^t$, and the current batch contributes $1-\beta_1$, so the share of first-moment

weight carried by earlier batches is

$$\frac{\sum_{s<t}(1-\beta_1)\beta_1^{t-s}}{1-\beta_1^t} = \frac{\beta_1 - \beta_1^t}{1-\beta_1^t}. \tag{18}$$

With $\beta_1 = 0.9$ it is 0, 47%, 63%, 71%, and 76% at updates 1–5 (the dashed line in Figure 2c) and approaches β_1 as t grows. It is a share of coefficients, not of displacement: the norms and alignment of the batch gradients determine how far each part actually moves the parameters.

Scope of the window. With $\beta_2 = 0.999$, the bias-corrected second moment at update $t \leq 5$ weights the t squared batch gradients almost equally, so D_t normalizes each coordinate by the root mean square of at most five batch gradients. A history share still rising toward β_1 and a coordinate scaling drawn from few batches both belong to the start of training. The analysis therefore describes how current and retained sources combine early in training; it does not cover longer runs, in which earlier batches carry about 90% of the first-moment weight and v_t averages over many more batches.

D Source Deletion, Matched Controls, and Full Results

This section tests whether local source rankings identify sets that affect actual action probabilities, and compares them with other sources at similar deletion scales. Cumulative first recovery measures how many sources are needed along a fixed ranking; controls at a fixed k test how well that ranking distinguishes effective sources. The two experiments share the target population and recovery definition but answer different statistical questions. Finally, we describe how often first-recovery sets contain historical sources, and what deleting only their current part can and cannot show.

D.1 Targets, Source Pools, and Finite Parameter Interventions

The target population contains all 10,636 records with nonzero credit and an observed change satisfying $r_{t,i} < 0$. For each target, the source pool contains every action occurrence from the first through the current update of the same seed, excluding only the current target record itself. Other occurrences of the same input, command, or task remain in the pool. We compute q using Eq. (4): strictly negative q defines opposing sources, strictly positive q defines supporting sources, and zero q belongs to neither group. Opposing sources are sorted by increasing q, with ties resolved by record identifier.

Let $V(S) = \sum_{(s,j)\in S} U_{s,j}^t$ be the summed vector of a source set S. Each intervention starts from the actual post-update parameters, replaces them with $\theta_t - V(S)$, and evaluates the target log-probability using its original input, response, and mask. Both U and D_t remain fixed across interventions. We neither run another optimization step nor recompute source gradients at the modified parameters. Recovery always uses the pre-update log-probability as its reference:

$$I_i(S) = \mathbf{1}\{\operatorname{sgn}(A_{t,i})[\ell_{t,i}(\theta_t - V(S)) - \ell_{t,i}(\theta_{t-1})] > 0\}. \tag{19}$$

Equality to zero does not count as recovery. For a negative-credit target, recovery means that its probability falls below the original pre-update value; for a positive-credit target, it must rise above that value. This definition does not use good/bad action labels.

D.2 Cumulative First Recovery along a Fixed Ranking

For target i, let $S_i(k)$ contain the first k sources in the negative-q ranking. We test $k = 1, \ldots, 32$ in order, or only the available prefixes if there are fewer sources. The first position satisfying $I_i(S_i(k)) = 1$ is denoted by k_i^*; targets that do not recover are assigned ∞. The main text calls $S_i(k_i^*)$ the target's recovering set; the appendices call it the first-recovery set. The cumulative fraction is

$$F(k) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\{k_i^* \leq k\}, \qquad N = 10,636. \tag{20}$$

A target that recovers at a smaller k is counted once and remains in the cumulative total. Testing stops at its first recovery, so $F(k)$ is not the fraction of all targets that remain recovered after deleting exactly k sources. Likewise, k_i^* is the first recovery position along this ranking, not the smallest set found by searching all subsets.

Per-target procedure.

1. Save the target log-probabilities before and after the actual update, and fix the source ranking.
2. Add the kth source to the prefix sum. Always subtract this sum from the actual post-update parameters, rather than subtracting the entire prefix again from the previous intervened model.
3. Evaluate the target with an actual forward pass. If strict recovery occurs, record $k_i^* = k$ and stop; otherwise continue.
4. Reset the actual post-update parameters afterward, and aggregate $\mathbf{1}\{k_i^* \leq k\}$ across targets.

Tables 14 and 15 report cumulative first recovery overall and by seed.

Table 14: Cumulative first recovery for all 10,636 targets.

Maximum sources removed	Targets recovered cumulatively	Cumulative fraction
1	5,028	47.27%
2	6,745	63.42%
4	8,572	80.59%
8	9,818	92.31%
16	10,495	98.67%
32	10,573	99.41%

Table 15: Cumulative recovered-target counts by seed. Every column uses the row's target count as its fixed denominator and includes all updates retained for that seed.

Seed	Targets	$k \leq 1$	$k \leq 4$	$k \leq 8$	$k \leq 32$
1	547	256	429	500	535
2	514	214	438	509	514
3	562	269	430	488	560
4	473	284	372	435	468
5	435	218	362	419	433
6	491	286	461	489	491
7	450	203	370	427	442
8	411	230	335	378	411
9	480	261	441	466	477
10	98	85	91	92	97
11	488	195	384	463	487
12	530	200	471	513	524
13	526	229	404	477	520
14	585	266	460	521	580
15	496	326	452	487	496
16	419	241	356	382	418
17	562	195	376	483	561
18	551	240	431	460	551
19	435	200	353	421	435
20	545	221	435	505	539
21	591	212	383	519	588
22	447	197	338	384	446

D.3 Controls at Fixed Source Counts: Matching Count and Summed Displacement Norm

At each fixed $k \in \{1, 4, 8\}$, we re-evaluate every available ranked prefix, including targets that previously recovered at a smaller k. The reference group uses the top k opposing sources. The other-opposing group samples from the remaining negative-q sources, excluding the target itself and all reference sources at that k; the supporting group samples positive-q sources and also excludes the target itself. Every set contains k distinct sources.

We match the Euclidean norm of the **summed vector**, $\|V(S)\|_2$, rather than $\sum_j \|U_j\|_2$. The former retains cross terms and cancellation within the set. A candidate must satisfy

$$0.9\|V(S_i(k))\|_2 \le \|V(S)\|_2 \le 1.1\|V(S_i(k))\|_2. \tag{21}$$

Candidate vectors are not rescaled. For each target, k, and control group, we sample at most 20,000 distinct sets. All combinations are enumerated when their total number is within this budget; otherwise, distinct sets are sampled uniformly. The random stream is determined by a fixed hash of the target identifier, k, and control group. Candidate generation and final selection precede all forward evaluations and do not use the recovery outcomes.

A target enters the common matched population $\mathcal{M}_k$ only if each control group has at least five distinct matched sets and the reference prefix contains k sources. If any condition fails, the target is excluded from the common comparison at that k. We neither relax the tolerance nor count a failed match as failed recovery. Each random control uses five sets, averaged within the target first:

$$\overline{I}_{i,a}(k) = \frac{1}{5}\sum_{r=1}^{5} I_i(S_{i,a,r}(k)), \qquad R_a(k) = \frac{1}{|\mathcal{M}_k|}\sum_{i\in\mathcal{M}_k} \overline{I}_{i,a}(k). \tag{22}$$

The reference group has one fixed ranked set per target and uses its recovery indicator directly. The version with equal weight for each seed first averages targets within each seed, then averages across seeds; the five random sets are not treated as five independent training runs. Because $\mathcal{M}_k$ may differ across k, changes in the plotted curves also reflect changes in the matched population.

Besides recovery, let $\ell_{t,i}(\theta) = \log \pi_\theta(a_{t,i} \mid h_{t,i})$. We record signed log-probability movement relative to the actual post-update model:

$$M_{t,i}(S) = \operatorname{sgn}(A_{t,i}) \left[\ell_{t,i}\left(\theta_t - \sum_{(s,j)\in S} U^t_{s,j}\right) - \ell_{t,i}(\theta_t) \right]. \tag{23}$$

A positive value indicates movement in the target's own credit direction, but does not necessarily cross its pre-update reference. For each random group, we again average the five sets within each target before summarizing the distribution across targets. This separates the size of a movement from whether it crosses the recovery threshold.

D.4 Completed Evaluations, Matching Coverage, and Recovery

All 106 updates and 10,636 targets were processed, with 93,478 forward evaluations under finite deletions. Before and after evaluation, we checked source identities, credit directions, source counts, actual deletion norms, and target probabilities after resetting the original parameters. Common matching coverage at $k = 1, 4, 8$ is 19.30%, 26.61%, and 11.98%, respectively. This coverage defines the scope of the controlled comparisons; unmatched records are not counted as recovery failures. Table 16 reports both equal-target and equal-seed results. The latter first aggregates the common matched targets within each seed, then gives equal weight to the 22 seeds.

Table 16: Recovery rates (%) at fixed k in the common matched population, weighting each target equally ("target") or first averaging within each seed and then weighting the 22 seeds equally ("seed"). Random controls first average five sets within each target. In every one of the 22 seeds and at every k, top opposing exceeds other opposing and other opposing exceeds supporting (Table 20); resampling seeds gives a 95% bootstrap interval of 82.2–89.1% for top opposing at $k = 4$.

		Top opposing		Other opposing		Supporting	
k	Common targets	target	seed	target	seed	target	seed
1	2,053	45.40	45.82	16.12	16.64	1.23	1.32
4	2,830	85.05	85.79	36.94	38.30	3.99	4.07
8	1,274	95.68	95.28	47.06	47.71	3.50	3.31

The matched population does not fully represent all targets. At the fixed values of k, the strongest opposing sources recover 47.27%, 80.09%, and 91.70% of all targets, compared with 45.40%, 85.05%,

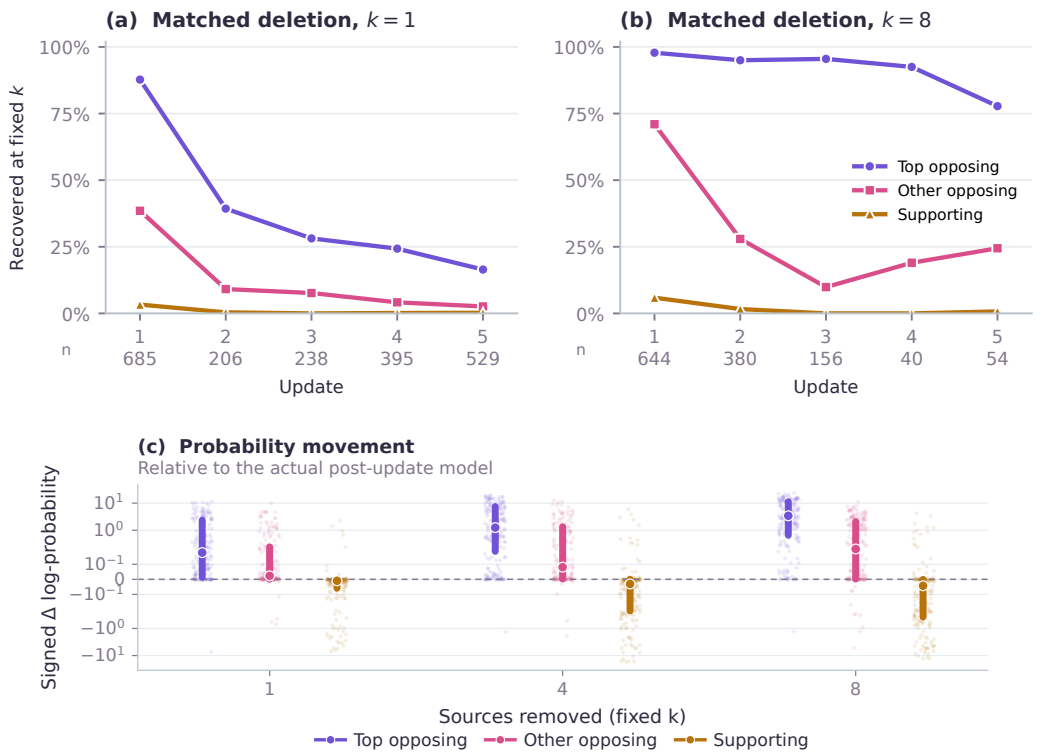


Figure 3: Full source-deletion results beyond Figure 2. **a, b**: Recovery at $k = 1$ and $k = 8$ by update on common matched targets (numbers under the axis; $k = 4$ is in Figure 2b). With a single deletion, the strongest opposing source recovers 87.7% of targets at update 1 but 16.5% at update 5; the $k = 8$ populations at updates 4 and 5 contain only 40 and 54 targets. **c**: Signed log-probability movement from the actual post-update model on a symmetric logarithmic scale; points and thick lines show medians and interquartile ranges, and faint points show a fixed sample. Random controls are first averaged within targets, and different values of k may use different target subsets.

and 95.68% in the common matched population. The corresponding rates outside that population are 47.72%, 78.29%, and 91.16%. These are re-evaluations at fixed k, not cumulative first-recovery rates. Tables 19 and 20 report results by update and seed, respectively.

Figure 3 reports the full movement distributions. Under Eq. (23), the median target movements for the strongest opposing sets at $k = 1, 4, 8$ are 0.177, 1.249, and 3.408. They are 0.023, 0.081, and 0.203 for other opposing sets, and -0.010, -0.031, and -0.043 for supporting sets. For random sets, the five outcomes are averaged within each target before computing the distribution across targets. These measurements retain changes that do not cross the recovery threshold.

D.5 Deletion Scale and Parameter Restoration

Norm matching compares the summed displacement of the three deletion sets; it does not require a deletion vector to be smaller than the original net update. Because sources can cancel within the original update, the norm of a removed contribution can exceed that of the update itself. The experiment therefore tests finite parameter changes under the actual fixed Adam scaling, not a small-perturbation approximation. Table 17 shows the norm of the strongest opposing deletion vector relative to the original update within the common matched population. For each target, the other two groups match this reference scale within $\pm 10\%$.

After each intervention, we reset the actual post-update parameters and re-evaluate the target; all logged restoration errors are zero. Norm matching controls intervention scale, but does not make

Table 17: Norm ratio of the strongest opposing deletion vector to the actual update on common matched targets.

k	Targets	Median	P90	P99
1	2,053	0.13	10.68	14.17
4	2,830	2.62	32.52	40.89
8	1,274	21.71	44.04	56.26

every geometric property of the sets identical. The conclusions apply to these matched source comparisons and do not imply the same outcomes from online data deletion or optimizer changes.

D.6 Historical Sources in First-Recovery Sets

We classify each reversed target by the origin of the sources in its first-recovery set $S_i = S_i(k_i^*)$: current only, mixed, or historical only, where historical sources come from earlier updates (Table 18). At updates 2–5, sets containing historical sources account for 22.9%, 59.7%, 69.6%, and 63.4% of reversed targets (4,412 of 8,216), and sets made only of historical sources for 11.0%, 19.3%, 34.2%, and 16.0% (1,648). For these targets, the strongest opposing contributions in the q ranking come partly or entirely from earlier batches. Through update 4 this share follows the weight that Adam's first moment places on past gradients (Appendix C.5); at update 5 it dips while that weight keeps rising. This composition does not depend on the deletion check below.

For mixed sets, we split S_i into current sources C_i and historical sources H_i and, keeping the actual post-update parameters and all source vectors fixed, delete only C_i. Full deletion already satisfies $I_i(S_i) = 1$, so the check records whether

$$I_i(C_i) = 0, \qquad I_i(C_i \cup H_i) = 1. \tag{24}$$

Current-only sets need no new forward pass ($C_i = S_i$). For history-only sets C_i is empty, so deleting it leaves the target reversed. We re-evaluated current-part deletion for all 2,764 mixed sets without changing the ranking, threshold, or targets, and took full-set outcomes from the frozen records. Re-evaluating one full set per update that contains a mixed set (81 checks) reproduces the frozen log-probabilities, and each parameter reset restores the target's original probability exactly.

Deleting only the current part recovers 121 of the 2,764 mixed sets and leaves 2,643 (95.6%) unrecovered. This outcome is largely fixed by construction. $S_i(k_i^*)$ is the shortest recovering prefix, so $S_i(k_i^* - 1)$ does not recover; when the last-ranked member of a mixed set is historical, C_i lies inside that failing prefix. Deletion along the ranking is nearly monotone: deleting the top k sources recovers 80.09% and 91.70% of all targets at $k = 4$ and 8 (Appendix D.4), against cumulative first recovery of 80.59% and 92.31% (Table 14), so a subset of a failing prefix is expected to fail as well. The same argument applies to deleting only H_i when the last-ranked member is current. The 95.6% therefore shows that mixed sets are close to minimal along the ranking, not that their historical part is more necessary than their current part.

Table 18: Composition of first-recovery sets by source origin, and the outcome of deleting only the current part of mixed sets. All sets in the set columns recover after full deletion. The unrecovered category contains targets not recovered within 32 sources, for which no first-recovery set exists. The five count columns sum to all reversed targets at that update; the last column gives mixed and history-only sets as a share of that total.

		Mixed sets: current part alone					Contains
Update	Current only	Recovers	Fails	History only	Unrecovered	Total	history (%)
1	2,408	0	0	0	12	2,420	0.0
2	1,591	22	228	230	25	2,096	22.9
3	793	46	762	386	13	2,000	59.7
4	616	15	707	700	6	2,044	69.6
5	753	38	946	332	7	2,076	63.4
Total	6,161	121	2,643	1,648	63	10,636	41.5

Counting mixed sets whose current part fails together with history-only sets, 4,291 targets at updates 2–5 satisfy Eq. (24): 52.23% of the 8,216 reversed targets, or 52.55% of the 8,165 recovered ones. By update, these rates are 21.85%, 57.40%, 68.84%, and 61.56% with equal target weight, and 22.41%, 56.98%, 72.09%, and 60.17% with equal seed weight. Figure 2c uses the equal-target rates, keeping the 51 targets without a first-recovery set in the denominator. These rates differ from the shares of sets containing historical sources by at most 2.3 points, and 1,648 of the 4,291 targets are history-only sets, which satisfy the event by definition, so the check adds little beyond the composition above.

We did not delete only H_i, compare against random partitions of S_i into parts of the same sizes, or separate mixed sets by the origin of their last-ranked member; these would test whether historical sources are distinctively required. Adding the historical part also enlarges the deleted set. The results therefore do not establish that historical sources are more necessary or more efficient than current ones, or that history is inherently harmful.

D.7 Matched-Control Results by Seed and Update

Tables 19 and 20 use the same common matching conditions as the main text. The update-level table pools targets at each update; the seed-level table pools the available updates of each seed. Each entry reports the number of common targets and the recovery rates of the three groups; the update-level table also gives the share of that update's reversed targets entering the common population. This share falls from 48.8% at update 1 to 12.5% at update 5 for $k = 4$, and is below 3% at updates 4 and 5 for $k = 8$, so later-update rates rest on a small, selected part of the reversals. No recovery rate is estimated where there are no common targets.

Table 19: Matched-control recovery rates (%) at fixed k, grouped by update. Random repetitions are averaged within each common target before equal-target aggregation. The last column gives common targets as a share of all reversed targets at that update (2,420, 2,096, 2,000, 2,044, and 2,076 at updates 1–5).

Update	k	Common targets	Top opposing	Other opposing	Supporting	Share of reversals
1	1	685	87.74	38.51	3.30	28.31
1	4	1182	97.97	65.09	8.66	48.84
1	8	644	97.83	70.99	5.90	26.61
2	1	206	39.32	9.13	0.39	9.83
2	4	559	88.73	26.55	1.18	26.67
2	8	380	95.00	27.95	1.63	18.13
3	1	238	28.15	7.65	0.00	11.90
3	4	453	72.85	9.23	0.04	22.65
3	8	156	95.51	9.87	0.00	7.80
4	1	395	24.30	4.15	0.15	19.32
4	4	377	75.07	14.80	0.74	18.44
4	8	40	92.50	19.00	0.00	1.96
5	1	529	16.45	2.61	0.23	25.48
5	4	259	54.05	11.58	0.31	12.48
5	8	54	77.78	24.44	0.74	2.60

Table 20: Matched-control recovery rates (%) at fixed k, grouped by seed. Each row uses only that seed's common matched targets at the given k.

Seed	k	Common targets	Top opposing	Other opposing	Supporting
1	1	157	36.94	7.90	1.02
1	4	170	74.71	26.00	1.76
1	8	56	92.86	40.36	3.57
2	1	105	46.67	10.86	1.52
2	4	139	83.45	29.21	1.29
2	8	40	92.50	40.00	1.50
3	1	164	43.29	20.49	0.85
3	4	192	69.27	34.90	1.46
3	8	53	83.02	47.55	2.26

Table20 (continued)

Seed	k	Common targets	Top opposing	Other opposing	Supporting
4	1	133	54.89	31.88	2.11
4	4	125	89.60	47.84	5.44
4	8	63	98.41	55.87	1.27
5	1	68	70.59	16.18	0.00
5	4	100	87.00	35.80	1.00
5	8	38	100.00	40.53	0.00
6	1	85	69.41	18.59	1.41
6	4	121	91.74	54.05	3.64
6	8	71	98.59	52.96	2.54
7	1	65	63.08	25.85	0.92
7	4	196	89.80	37.45	3.88
7	8	120	100.00	47.67	3.33
8	1	39	38.46	6.15	0.00
8	4	109	82.57	42.94	7.52
8	8	42	97.62	72.86	1.43
9	1	100	69.00	36.60	2.80
9	4	189	91.53	45.29	9.42
9	8	104	93.27	45.96	6.15
10	1	21	85.71	62.86	6.67
10	4	48	97.92	89.58	6.67
10	8	34	97.06	91.76	6.47
11	1	71	54.93	13.80	1.41
11	4	93	83.87	18.28	1.51
11	8	46	97.83	42.61	3.48
12	1	70	25.71	2.86	0.29
12	4	157	95.54	20.64	1.53
12	8	49	97.96	42.45	1.22
13	1	106	41.51	19.25	2.08
13	4	140	88.57	46.43	5.43
13	8	53	96.23	39.25	6.79
14	1	132	23.48	4.24	0.15
14	4	87	90.80	37.93	2.99
14	8	37	89.19	49.19	3.78
15	1	113	64.60	33.98	3.54
15	4	156	89.74	44.74	8.46
15	8	86	100.00	53.95	9.30
16	1	32	18.75	1.25	0.00
16	4	54	98.15	29.63	5.56
16	8	33	100.00	35.15	0.61
17	1	109	35.78	4.59	0.37
17	4	103	90.29	49.71	5.44
17	8	52	88.46	57.31	5.38
18	1	126	46.83	11.75	0.48
18	4	115	81.74	25.57	1.22
18	8	27	100.00	32.59	0.00
19	1	39	10.26	4.10	0.51
19	4	137	80.29	45.26	6.57
19	8	79	98.73	47.09	2.53
20	1	127	51.18	21.26	0.94
20	4	172	88.95	27.79	0.23
20	8	96	92.71	26.25	1.04
21	1	114	24.56	3.68	0.53
21	4	110	70.91	35.82	7.27
21	8	30	83.33	42.67	8.00
22	1	77	32.47	8.05	1.56
22	4	117	70.94	17.78	1.37
22	8	65	98.46	45.54	2.15

E Auditing Source Experiences: Relations and Complete Cases

Source deletion establishes how a set changes a target's actual response. This section examines what the records in that set were doing at the time, and how their actions relate to the target. These pairs are selected through reversal events and first-recovery sets, so the relation statistics describe the behavioral meaning of the identified effects rather than of random action pairs; Appendix B.2 states their units and scope.

E.1 Audit Units and Scope

For every target recovered within 32 sources, we take the complete prefix at its first strict recovery along the fixed negative-q ranking. Each member of this prefix forms a source–target record. Unrecovered targets retain that status, without assigning them recovery sources. Recovery of a set does not establish that every member is individually necessary. We call a pair same-sign if the source and target have total credit of the same sign, and opposite-sign otherwise. All sources considered here have negative q: same-sign credit does not mean a supporting effect on the target.

The first-update audit covers seeds 1–20 and contains no optimizer history; seeds 21 and 22 were not audited at the first update, so cross-stage comparisons use only seeds common to both audits. Its opposite-sign statistics include 1,711 raw pairs involving 1,516 targets. The more detailed same-sign audit contains 944 deduplicated inputs, corresponding to 1,056 raw occurrences in its frozen input set. The later audit covers updates 2–5 of seeds 1–9 and 11–22: 84 target updates and 8,216 reversed targets, of which 8,165 recovered within a prefix and 51 did not. Their recovery sets contain 28,318 pairs (14,220 same-sign and 14,098 opposite-sign). Historical records are read with the task and state at the time of the source; the target's current state is not substituted for the source's past state.

The cross-stage comparison in the main text uses the 19 seeds common to both audits, namely 1–9 and 11–20, and counts raw source–target occurrences. Within this scope, the first update contains 979 same-sign pairs and 1,620 opposite-sign pairs. Deduplication is used only to reuse inference for identical inputs; it does not remove a record's occurrence weight in actual training. We therefore report the 20-seed first-update table based on consensus across both input orders separately from the 19-seed update-level table based on raw records, rather than combining them into one overall rate.

E.2 Relations: Separating Operations, Effects, and Task Roles

The environment's four commands have deterministic physical effects, but the same command can play different roles depending on task progress. Table 21 defines the comparisons used in the audit. Each relation is judged separately and relations may overlap. Having the same task role does not imply the same long-term value, nor does it follow from both actions being judged reasonable.

Operation type is ultimately computed directly from the command. Shared direct effects are also checked uniformly using environment rules: `take` moves an object into the hand, `put/drop` frees the hand, and `open` opens a container. Among legal actions in this environment, matching operation types necessarily share a basic effect, while the cross-operation pair that shares an effect is `put/drop`. Goal completion and reversal remain part of task role. Sharing at least one effect is not equality of the entire state change. Thus, matching operation type and sharing an effect do not provide two independent measures of explanatory coverage.

Judging state and task role requires interpreting objects and containers within the task, for which we use frozen model judgments. Names may be mapped by task role, but the mapping must remain consistent throughout a comparison; it cannot change from one condition to another. For example, an unfinished goal object on the table and one in an open container are both directly accessible, whereas an object in a closed container still requires opening it. The same command may also have different task roles when its object is unfinished in one record and already completed in the other.

E.3 Model Inputs, Prompts, and Blinding

The audit uses Qwen3.8-27B-FP8 (Qwen Team, 2026b) at temperature 0 with thinking enabled. Each record provides the task, complete pre-action state, actual action, and the preceding experience needed for the judgment. The similarity audit includes within-trajectory history and remaining

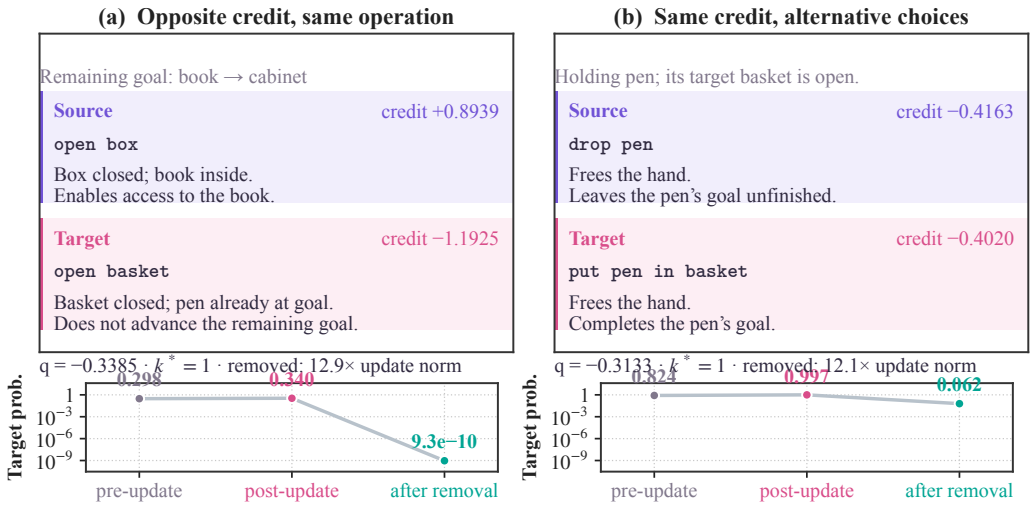


Figure 4: Verified source relations at the first update. **a**: Opening in different task roles with opposite credit (seed 1, case A). **b**: Placing and dropping with same-sign credit in the same physical state but after different histories: both free the hand but serve different purposes, consistent with competition between alternatives (seed 3, case B). Each source alone restores its target ($k^* = 1$); probabilities (log scale) show pre-update, post-update, and deletion states. The removed contributions, 12.9 and 12.1 times the norm of the actual update, are finite interventions magnified at the first update, so the cases illustrate the two relations rather than typical effect sizes.

steps; the choice-relation audit additionally supplies legal commands, environment-computed action changes, and candidate cross-mappings for inspection. The records describe independent experiences, not successive parts of one trajectory. Inputs omit credit values and signs, gradients or q, pre- and post-update probabilities, recovery outcomes, good/bad labels, and source age in optimizer history. These attributes are joined after inference. In the prompt, "similar history" refers to actions already taken within a trajectory, not whether the records originate from historical updates.

The similarity prompt is identical in the first update and updates 2–5, returning four judgments: history, current state, action, and task role. The detailed same-sign audit uses a seven-field prompt in the first update, of which five fields are retained in the reported statistics. Updates 2–5 use a five-field version that removes prerequisite–execution and complementary prerequisites. These are not described as identical inference conditions; both full versions appear in Appendix G. The original first-update outputs were not regenerated with the five-field prompt.

Constructing inputs for alternative choices. We hold the actual actions of A and B fixed. The program lists legal commands in A that have the same operation type as B's actual action but differ from A's actual action, and does the symmetric construction in B. It enumerates and numbers their cross-combinations. The model checks whether a candidate consistently maps "A's other command" to "B's actual action" and "B's other command" to "A's actual action," preserving object roles, task roles, and object relations. A positive judgment must identify a passing candidate; a negative judgment requires that none passes. The model therefore cannot construct an alternative-choice relation by replacing an actual action or importing the source state into the target record.

The initial output budgets for the later similarity and choice-relation requests are 8,192 and 16,384 tokens, respectively. Technical failures, empty responses, and truncations are retried with the same input; some incomplete requests receive an increased budget of 24,576. Retrying depends on response completeness, not whether the relation judgment meets an expectation. All original-order and prescribed swapped-order outputs were eventually completed. Technical completion does not establish semantic correctness.

Table 21: Criteria for audit relations. Other necessary conditions in these examples are assumed to be supplied and satisfied; the actual audit uses the complete input facts.

Relation	What is compared	Distinguishing examples
Similar history	Previous operation patterns, their order, and prerequisite dependencies	Repetition counts may differ once both histories contain the same complete round trip; no round trip and an existing round trip differ
Similar current state	Remaining goals, held objects, prerequisites for taking and placing, and relevant budget, under a task-role mapping	An empty hand and a directly accessible book/cup can be similar; needing to free the hand differs from needing to open a box
Same operation type	The actual command type: `take/put/drop/open`	Two `put` commands can match; `put/drop` differ
Shared direct effect	Whether at least one immediate state change is shared	`put/drop` both free the hand; this does not make their full state changes equivalent
Same task role	Object roles and changes in goal completion/reversal, holding, and required access	Taking an unfinished goal object differs from removing a completed one; two ways to set down an irrelevant object can match
Alternative choices	With actual actions fixed, distinct legal choices can be mapped by role in both original states	When holding a goal object with its destination open, correct placement and dropping can be alternatives; do not invent a command unavailable in the other state
Opposite task progress	One action completes a goal and the other reverses a completed goal with a corresponding role	Taking an unfinished object does not complete its goal; dropping a held object does not automatically reverse a completed goal

E.4 Order Checks, Rule Checks, and Disagreements

The similarity prompt was developed using first-update seeds 1–5 and then frozen before application to seeds 6–20 and later updates. For the first update, approximately one tenth of the requests were selected by request hash for an order swap. The original order remains the reported result and is not automatically corrected using the swapped output. The logs contain 255 swap checks, or 232 unique input pairs, of which 6 disagree on at least one field. This check draws from all inputs to the four-question audit; it is not an accuracy estimate for the opposite-sign subset.

All 944 deduplicated same-sign inputs in the first update were evaluated in both orders. A relation is recorded as present only if both orders affirm it, absent only if both reject it, and unresolved otherwise. Alternative choices have 533 present, 385 absent, and 26 unresolved judgments; same task role has 156 present, 786 absent, and 2 unresolved. A total of 27 inputs disagree on at least one retained field. Counting every unresolved case as absent or present gives ranges of 56.46%–59.22% for alternative choices and 16.53%–16.74% for same task role. These are bounds under different treatments of disagreements, not confidence intervals.

All later same-sign inputs were evaluated in swapped order, while opposite-sign inputs were selected for swaps using a fixed hash sample. The original order is the prespecified primary result; disagreements are neither removed nor repaired by majority voting. Table 22 reports the checks by unique input for updates 2–5 across all 21 seeds. Several disagreeing fields can belong to the same input, so summing field counts does not give the number of erroneous records.

The program also checks operation type and shared effects, which can be determined directly from commands. For all 979 first-update same-sign records in the 19 common seeds, the shared-effect rule agrees with the model's original judgment; there is 1 disagreement in the corresponding later-update scope. The complete later-update material for 21 seeds also retains 13 unique inputs flagged for possible rule violations, representing 19 raw occurrences. These mainly concern judgments that did not compare operations by command type. Operation and effect statistics in the main text use uniform environment rules; state and task-role judgments retain the frozen original-order outputs.

Table 22: Order-swap checks for updates 2–5, counted by unique original-order input. For 16 opposite-sign inputs, swapping leaves the content unchanged. A disagreement means at least one field differs.

Protocol	Inputs checked	Changed by swapping	Inputs with disagreements
Same-sign, five fields	7567	7567	436
Opposite-sign, four questions	996	980	56

Table 23: Semantic audit of first-recovery source sets in the first update across 20 seeds. Opposite-sign entries count raw source–target records; same-sign entries count deduplicated annotation inputs. Relations may overlap, and rates with different denominators cannot be combined.

Credit relation and unit	Audited relation	Present/total	Rate
Opposite-sign: raw records	Similar action	1,649/1,711	96.38%
	Similar current state	531/1,711	31.03%
	Similar task role	963/1,711	56.28%
Same-sign: unique inputs	Alternative choices	533/944	56.46%
	Alternative choices with a shared direct effect	453/944	47.99%
	Same operation type	252/944	26.69%
	At least one shared direct effect	629/944	66.63%
	Same task role	156/944	16.53%
Same-sign: confirmed alternatives	Different task roles	407/533	76.36%

For same-sign entries, a relation is present only if both presentation orders agree. There are 26 order disagreements for alternative choices and 2 for the same task role, all retained in the denominator of 944. Together, these fields involve 27 distinct inputs; field disagreements are not summed as sample counts. Opposite-sign entries use frozen original-order results and sampled swap checks; see Appendix E for the protocol and Appendix B.2 for units and scope.

These checks assess order stability and some explicit rules, but do not provide an accuracy estimate based on independent human annotation.

E.5 Full Relation Distributions and Weighting

Table 23 retains the first-update results for 20 seeds to clarify the distinction between alternative choices and task role. Opposite-sign entries count raw pairs; same-sign entries count deduplicated inputs with consensus across both orders. Of the 533 inputs judged to offer alternative choices, 407 have different task roles. Sharing a direct effect likewise does not imply serving the same purpose in the task.

Cross-stage descriptions use the 19 common seeds and raw occurrence counts. Table 24 gives counts of pairs satisfying each relation and their denominators for opposite-sign relations in updates 2–5; Table 25 gives mutually exclusive cross-tabulations of operation and effect for same-sign pairs. In the first update, 25.8% of same-sign pairs share an operation type, while another 37.6% have different operation types but share a direct effect. By update 5, the latter category accounts for 0% of current and 2.8% of historical pairs. This cross-tabulation distinguishes `put/drop` from `put/put` and `take/take`, rather than treating two correlated percentages as separate discoveries.

Table 24: Raw opposite-sign pair counts for the 19 common seeds. Dividing each of the last three columns by the row's pair count gives the corresponding rate in the main text. The three relations may overlap.

Update	Source	Pairs	Same operation	Similar state	Similar task role
2	Current	1573	1278	463	877
2	Historical	407	402	61	205
3	Current	2250	2194	719	1557
3	Historical	950	902	194	551
4	Current	1665	1475	521	917
4	Historical	1351	1228	244	620
5	Current	2283	2049	641	1301
5	Historical	1602	1491	319	839

Table 25: Cross-tabulated same-sign pair counts for the 19 common seeds. The last three columns are mutually exclusive and sum to the row's pair count. No legal command pair has the same operation type without sharing a basic effect.

Update	Source	Pairs	Same operation and effect	Different operation, shared effect	Neither
1	Current	979	253	368	358
2	Current	2044	1263	122	659
2	Historical	388	212	53	123
3	Current	1597	1114	126	357
3	Historical	1228	777	96	355
4	Current	1307	1114	1	192
4	Historical	1841	1168	144	529
5	Current	2278	1572	0	706
5	Historical	1851	1230	52	569

E.6 Candidate Baselines: Are Selected Relations Already Common?

We construct separate candidate pools for opposite-sign and same-sign sources. For each target, current candidates are all other nonzero-credit records in the same batch whose credit has the opposite sign (or the same sign); historical candidates are all records from earlier updates of the same seed satisfying that sign condition. Candidates are not filtered by q or recovery outcome. We retain actual occurrences and never add records from other seeds. For either sign relation and source category $c \in \{\text{current}, \text{historical}\}$, let $S_{i,c}$ denote the corresponding sources in the first-recovery set and $P_{i,c}$ all their candidates. In the opposite-sign comparison, $R(i,j)$ indicates the same operation type; in the same-sign comparison, it indicates a shared direct effect. Both are computed from environment command rules. Define

$$C_{i,c} = \frac{1}{|S_{i,c}|} \sum_{j \in S_{i,c}} R(i,j), \qquad B_{i,c} = \frac{1}{|P_{i,c}|} \sum_{j \in P_{i,c}} R(i,j). \tag{25}$$

We average $C_{i,c}$ and $B_{i,c}$ separately only over targets with nonempty $S_{i,c}$, giving Table 1 in the main text; their candidate pools are then also nonempty. The current and historical columns can involve different targets, and the first update has no historical column. This comparison measures concentration of the selected relation relative to available candidates, not a causal effect of historical versus current sources on the same target.

Equal-target weighting and raw pair counts answer different questions. In the first update, shared effects account for 63.4% of pair records, whereas the mean proportion among each target's selected sources is 66.5%. The former gives more weight to targets with more sources. Table 26 reports target counts, selected-pair counts, and their comparisons to keep these rates distinct. This baseline controls candidate composition, but does not match credit magnitude, gradient norm, or source age.

Table 27 reports the equal-target comparison for matching operations with opposite-sign credit. It uses the same 19 seeds and weighting as the main-text same-sign table. In updates 2–5, matching operations account for 29.3%–36.6% of historical candidates, compared with 91.4%–97.8% of

Table 26: Shared-effect candidate comparisons and denominators. Percentages are computed per target and then averaged with equal target weights; they differ from percentages obtained by pooling pair counts.

Update	Source	Targets	Selected pairs	Candidate (%)	Selected (%)
1	Current	750	979	41.6	66.5
2	Current	879	2044	36.9	60.1
2	Historical	294	388	38.0	73.6
3	Current	811	1597	37.9	76.6
3	Historical	556	1228	37.9	70.8
4	Current	629	1307	41.0	84.0
4	Historical	862	1841	38.8	72.9
5	Current	1052	2278	37.7	70.8
5	Historical	820	1851	39.8	69.5

Table 27: Full counts for the opposite-sign, same-operation candidate comparison over 19 common seeds. Candidate and identified-source rates are computed per target and averaged with equal weights. Seed counts include only seeds with targets of that category.

Update	Source	Targets	Seeds	Selected pairs	Candidate same-op.	Selected same-op.
1	Current	1,437	19	1,620	30.6%	97.2%
2	Current	961	19	1,573	35.5%	78.8%
2	Historical	225	17	407	29.3%	97.8%
3	Current	931	19	2,250	37.5%	95.9%
3	Historical	547	19	950	31.3%	95.0%
4	Current	885	16	1,665	41.5%	86.1%
4	Historical	695	19	1,351	36.2%	91.4%
5	Current	1,120	19	2,283	39.4%	88.9%
5	Historical	745	19	1,602	36.6%	92.7%

identified sources. The relation's prevalence therefore cannot be explained solely by its prevalence among all available sources. Table 24 separately reports state and task-role distributions; matching operations cannot stand in for those relations.

Table 28 first computes the two target-averaged proportions within each seed and then averages across seeds, checking whether seeds with more targets dominate the result. For current sources in update 2, the selected proportion of matching operations exceeds the candidate proportion in 18/19 seeds. It is higher at every other reported position among seeds with eligible targets. The main text uses the equal-target values (Appendix B.2).

Table 28: Equal-seed check of the opposite-sign, same-operation comparison. Targets are first weighted equally within each seed, then seeds with that category of target are averaged. The last column counts seeds whose selected rate exceeds their candidate rate. This is a descriptive check across seeds, not an analysis treating pairs from the same batch as independent replicates.

Update	Source	Candidate same-op.	Selected same-op.	Seeds: selected above candidate
1	Current	30.3%	97.3%	19/19
2	Current	35.8%	79.3%	18/19
2	Historical	31.7%	98.3%	17/17
3	Current	37.4%	95.9%	19/19
3	Historical	31.6%	96.1%	19/19
4	Current	41.5%	88.6%	16/16
4	Historical	36.7%	93.1%	19/19
5	Current	38.8%	86.6%	19/19
5	Historical	36.5%	92.3%	19/19

E.7 Case A: Opposite Credit for the Same Opening Operation

Figure 4a comes from the first update of seed 1. Both records have the same goals: place the cup in the box, the book in the cabinet, and the pen in the basket. Before the actions, object locations are identical: the apple is in the cabinet, the book and cup in the box, the key on the table, and the pen and bottle in the basket; the hand is empty. The cup and pen goals are complete, while the book must still move from the box to the cabinet. Table 29 places the actual differences side by side.

Table 29: Source and target in case A. The records belong to different sampled trajectories; the source action does not follow the target action in one trajectory.

	Target record	Source record
Previous commands	`put apple in cabinet`	1. `put apple in cabinet` 2. `open basket`
Container states	Box and basket closed; cabinet open	Box closed; basket and cabinet open
Remaining steps	13	12
Actual action	`open basket`	`open box`
Total credit	-1.1924784	$+0.8938914$
Direct task relation	Opens the basket containing the completed pen goal	Opens the box so the unfinished book can be taken

Both commands open a container. The source opens the container needed to access the book; opening the target's basket does not supply a prerequisite for the remaining book goal. The frozen judgments are similar actions, similar current states, different histories, and different task roles. The source has $q = -0.3385258$ and ranks first; its single contribution is the first-recovery set. The target probability rises from 0.2979624 to 0.3400467, against its negative credit. After deletion it is 9.31×10^{-10}, restoring the negative-credit direction. The deletion norm is 12.88 times the actual net-update norm, so this is a finite parameter intervention, not a small perturbation.

This case connects opposite-sign credit, matching operations, and an actual deletion outcome. It does not establish whether opening the basket should itself become more or less probable, or whether the same operation always produces the same gradient relation across states. The internal case identifier is `S01T016R01`; all facts needed for the explanation and calculation are given here and do not require an external file indexed by that identifier.

E.8 Case B: Two Choices Free the Hand but Serve Different Task Roles

Figure 4b comes from the first update of seed 3, again with the goals of placing the cup in the box, the book in the cabinet, and the pen in the basket. The physical states before the two actions are identical: the apple is in the cabinet, the book and cup in the box, the key on the table, the pen in hand, and the bottle in the basket. All three containers are open. The cup goal is complete, while the book and pen goals remain unfinished. Table 30 gives both records' complete action histories.

Table 30: Complete action histories and actual choices in case B. Identical locations do not imply identical histories in the model inputs.

History position	Target record	Source record
1	`open basket`	`put apple in cabinet`
2	`put apple in cabinet`	`take apple`
3	`take key`	`put apple in cabinet`
4	`put key in basket`	`open basket`
5	`take key`	`take pen`
6	`drop key`	`open box`
7	`open box`	History ends at entry 6
8	`take pen`	
Remaining budget	6 steps	8 steps
Actual action	`put pen in basket`	`drop pen`
Total credit	-0.4019633	-0.4162976

Both placing the pen in the basket and dropping it on the table are executable in each record's original state. Both free the hand, but only placing it in the basket completes the pen goal. Both input orders confirm alternative choices and a shared direct effect, but reject the same task role. Here, "alternative" means two concrete choices that can be mapped across the records, not choices equivalent for the task.

The source has $q = -0.3132890$ and also ranks first. Despite its negative credit, the target's probability rises from 0.8242481 to 0.9970097. Deleting this source lowers it to 0.0619636, restoring the negative-credit direction. The deletion norm is 12.07 times the actual net-update norm. The source's penalty on dropping and the target's penalty on placing have the same credit sign, yet their cross-effect relative to the target's credit direction is negative, consistent with competition between related choices. This case is indexed as `S03T013R01`; the single source is itself the first-recovery set, so its evidence is not borrowed from the validation of a different, multi-source set.

The two cases illustrate complete evidence chains, while the full relation statistics and deletion controls establish their scope. Sources not covered by these relations remain unexplained; we do not call them random effects. Both cases come from the first update and lie in the upper tail of deletion scale: their removed contributions are 12.88 and 12.07 times the update norm, whereas among common matched targets at $k = 1$ a single-source deletion has a median of 0.13 times and a 90th percentile of 10.68 times the update norm (Table 17). They illustrate the two relations rather than typical deletions; the matched controls in Appendix D describe typical behavior.

F History Retention in ALFWorld and WebShop: Setup and Full Results

F.1 Training Setup

The retention experiment uses the GiGPO and GRPO recipes in `verl-agent` for ALFWorld and WebShop, trains all parameters of Qwen2.5-1.5B-Instruct, and adds only the history adjustment of Eq. (6). Within each algorithm, environment, and learning rate, `original` and `cap2` share every other setting and the environment seed (seed 0). In addition to the settings summarized in Section 4.2, GiGPO on WebShop was also trained at 4×10^{-6}, and `cap2` was repeated with seed 1 for GiGPO on WebShop at 10^{-6}; these runs appear only in this appendix. A training iteration collects one batch of rollouts and may contain several optimizer updates, so iterations are neither the optimizer steps t of Eq. (6) nor environment steps. Validation at iteration 0 and every 50 iterations uses 128 sampled validation episodes; it is a sampled estimate rather than a pass over the split, with a standard error of about 4 points for success rates between 20% and 80%. The iteration-200 checkpoints of the seed-0 runs are also evaluated on complete evaluation sets (Appendix F.4).

For each optimizer update, we first backpropagate the actual total loss and apply the existing gradient clipping. We then use g_t to calculate the second moment $v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^{\odot 2}$ that will enter AdamW and its bias correction, obtaining $D_t = \mathrm{diag}((\sqrt{v_t/(1-\beta_2^t)}+\epsilon)^{-1})$. Historical and current norms are computed jointly over all trainable parameters participating in the update. Because the total gradient includes policy, KL, and other enabled losses, H_t and C_t are not entirely action-credit contributions. The FSDP implementation sums the squared norms across shards before computing a common ρ_t. Each parameter's `exp_avg` is multiplied in place by ρ_t before `optimizer.step()`; the first update is unchanged because it has no history. Updates with non-finite gradients are skipped according to the existing logic. The recurrence and its expansion are

$$\begin{aligned} m_t &= \beta_1 \rho_t m_{t-1} + (1-\beta_1) g_t, \\ &= (1-\beta_1) \sum_{r=1}^{t} \beta_1^{t-r} \left(\prod_{k=r+1}^{t} \rho_k \right) g_r. \end{aligned}$$

The empty product is 1, and all gradients in this expression are those actually encountered by the intervention arm. Past signals are attenuated again by every later cap. Since both the parameters and sampled data change, this expression does not describe an entire training trajectory obtained by reweighting the baseline's fixed gradient sequence. `original` always uses $\rho_t = 1$, and `cap2` uses $\kappa = 2$.

F.2 Effective Step Length under Attenuation

The bias correction $1-\beta_1^t$ does not depend on ρ_t, so attenuating m_{t-1} changes the length of the update as well as its direction. Two limiting cases make this explicit. If successive gradients are uncorrelated with equal norm, $\mathbb{E}\|m_{t-1}\|_2^2 \approx \frac{1-\beta_1}{1+\beta_1}\|g\|_2^2$ in steady state, so $\|H_t\|_2/\|C_t\|_2 \approx \beta_1/\sqrt{1-\beta_1^2} \approx 2.06$ for $\beta_1 = 0.9$ (ignoring differences in D_t). A threshold of $\kappa = 2$ therefore sits at the ratio produced by ordinary gradient noise, and `cap2` can clip updates throughout training, not only when history dominates. If instead successive gradients are aligned, the unmodified moment approaches g and the history term is about nine times the current term. A cap limits the history term to κ times the current term, so the step is at most $(1+\kappa)(1-\beta_1) = 0.3$ of the unmodified step for $\kappa = 2$. Attenuation therefore shortens steps most in phases where successive gradients agree.

F.3 All Validation Values

Table 31 lists every validation value of the experiment, with the difference `cap2` minus `original` at each checkpoint of a seed-0 pair. At iteration 0 both arms evaluate the same initial model, so the difference is zero; the seed-1 run starts from a different value because validation sampling depends on the seed. At iteration 200 the seven differences range from -6.3 to $+6.3$ points, three positive and four negative, all below two standard errors of a difference (about 11 points; Appendix F.5). At iterations 50–150, four gaps exceed this threshold at 10^{-6}, all favoring `original`: -14.9 (GiGPO–WebShop, iteration 50), -13.3 and -18.8 (GRPO–ALFWorld, iterations 100 and 150), and -12.5 (GRPO–WebShop, iteration 50). At 2×10^{-6}, two exceed it, in opposite directions: $+13.3$ (GRPO–ALFWorld, iteration 150) and -35.1 (GRPO–WebShop, iteration 50), the latter closing by iteration 150. At 4×10^{-6} (GiGPO–WebShop only), one exceeds it: -11.7 at iteration 100. Without a threshold, `cap2` trails in seven of nine early comparisons at 10^{-6}, three of nine at 2×10^{-6} (with two ties), and two of three at 4×10^{-6}; the three checkpoints of a run pair are not independent comparisons.

The seed-1 repeat of `cap2` (GiGPO–WebShop, 10^{-6}) differs from the seed-0 `cap2` run by -11.7, $+14.8$, -1.6, and $+0.8$ points at iterations 50–200. Two runs of the same arm can thus differ as much as the early gaps between arms at 10^{-6} (12.5–18.8 points), so those gaps cannot be separated from run-to-run variation. No seed-1 run of `original` exists, so this repeat enters no difference.

Table 31: Validation success (%) on 128 sampled episodes at iterations 0–200, and the difference `cap2` minus `original` (percentage points). All runs use seed 0 unless marked.

Setting	LR	Arm	0	50	100	150	200
GiGPO, WebShop	10^{-6}	`original`	4.7	38.3	61.7	63.3	76.6
		`cap2`	4.7	23.4	51.6	58.6	70.3
		difference	0.0	−14.9	−10.1	−4.7	−6.3
		`cap2`, seed 1	2.3	11.7	66.4	57.0	71.1
	2×10^{-6}	`original`	4.7	39.8	59.4	56.2	68.8
		`cap2`	4.7	48.4	55.5	62.5	73.4
		difference	0.0	+8.6	−3.9	+6.3	+4.6
	4×10^{-6}	`original`	4.7	42.2	71.9	60.2	71.1
		`cap2`	4.7	45.3	60.2	58.6	69.5
		difference	0.0	+3.1	−11.7	−1.6	−1.6
GRPO, ALFWorld	10^{-6}	`original`	7.0	18.8	46.1	68.0	73.4
		`cap2`	7.0	20.3	32.8	49.2	77.3
		difference	0.0	+1.5	−13.3	−18.8	+3.9
	2×10^{-6}	`original`	7.0	22.7	44.5	60.9	73.4
		`cap2`	7.0	22.7	51.6	74.2	79.7
		difference	0.0	0.0	+7.1	+13.3	+6.3
GRPO, WebShop	10^{-6}	`original`	4.7	30.5	45.3	58.6	64.1
		`cap2`	4.7	18.0	47.7	54.7	61.7
		difference	0.0	−12.5	+2.4	−3.9	−2.4
	2×10^{-6}	`original`	4.7	52.3	57.0	55.5	64.8
		`cap2`	4.7	17.2	50.0	55.5	61.7
		difference	0.0	−35.1	−7.0	0.0	−3.1

F.4 Full Evaluation of the Final Checkpoints

Table 32 evaluates the iteration-200 checkpoints of the seed-0 runs on complete evaluation sets: the 140 seen and 134 unseen ALFWorld games and 500 WebShop tasks. Across the nine comparisons, `cap2` minus `original` ranges from -10.6 to $+9.4$ points and is positive in five. Two differences exceed two standard errors, in opposite directions: $+9.4$ for GiGPO–WebShop and -10.6 for GRPO–WebShop, both at 2×10^{-6}. Taking the better learning rate for each arm, `cap2` leads by 4.8 points on GiGPO–WebShop (77.4% versus 72.6%), trails by 6.2 points on GRPO–WebShop (64.0% versus 70.2%), and leads by 2.1 (seen) and 4.5 (unseen) points on GRPO–ALFWorld. The full evaluation therefore agrees with validation that uniform attenuation has no consistent effect on final success. It also shows how far a 128-episode validation can depart from a complete evaluation: for GRPO–WebShop at 2×10^{-6} the validation difference is -3.1 points and the full-evaluation difference -10.6, and for GiGPO–WebShop at 4×10^{-6} they are -1.6 and $+4.4$. The standard errors cover evaluation sampling only; variation between training runs (Appendix F.3) comes on top of them.

Table 32: Success (%) of the iteration-200 checkpoints (seed 0) on complete evaluation sets, with solved counts in parentheses, the difference `cap2` minus `original` (percentage points), and its binomial standard error.

Setting	LR	Evaluation set (n)	`original`	`cap2`	Difference	SE
GiGPO, WebShop	10^{-6}	full (500)	72.6 (363)	71.4 (357)	-1.2	2.8
	2×10^{-6}	full (500)	68.0 (340)	77.4 (387)	$+9.4$	2.8
	4×10^{-6}	full (500)	71.8 (359)	76.2 (381)	$+4.4$	2.8
GRPO, ALFWorld	10^{-6}	seen (140)	75.0 (105)	77.1 (108)	$+2.1$	5.1
		unseen (134)	71.6 (96)	74.6 (100)	$+3.0$	5.4
	2×10^{-6}	seen (140)	74.3 (104)	73.6 (103)	-0.7	5.2
		unseen (134)	69.4 (93)	76.1 (102)	$+6.7$	5.4
GRPO, WebShop	10^{-6}	full (500)	67.6 (338)	64.0 (320)	-3.6	3.0
	2×10^{-6}	full (500)	70.2 (351)	59.6 (298)	-10.6	3.0

F.5 Scope of the Retention Evidence

Apart from one seed-1 repeat of `cap2`, each configuration was run once, so every comparison rests on a single pair of runs and run-to-run variation is measured only for that repeat. With 128 validation episodes, differences below about 11 points (two standard errors of a difference) should not be read as effects; on the complete evaluation sets the corresponding thresholds are about 6 points for WebShop and 10–11 points for ALFWorld, before run-to-run variation. We did not test a variant that renormalizes the attenuated moment to preserve step length, nor one that attenuates only history components opposing the current gradient; separating the effect of direction from that of step length requires such controls. The experiment is independent of the controlled source analysis in Section 3 and does not trace individual sources.

G Audit Prompts

This appendix reproduces the three system prompts used in the experience audit of Appendix E.

G.1 Full Prompt for the Four-Question Similarity Audit

The following is the system prompt shared by the first-update and updates 2–5 audits, with all rules and examples retained. The user message contains the two records' tasks, actual histories, complete current states, environment-computed goal progress and held objects, remaining steps, and actual actions. The output contains the four prescribed fields and does not require a behavioral explanation. *Translated from the original prompt.*

Four-question similarity audit

```
Compare two action records from text-based manipulation tasks and judge each of
the following separately:

1. Whether their histories are similar;
2. Whether their current states are similar;
3. Whether their actions are similar;
4. Whether the actions have similar task roles.

Each record contains task goals, previous actions, the current state, the
present action, and possibly the number of remaining steps. Goal progress and
held-object information in the input are computed directly from current object
locations. They are environment facts, not evaluations of whether an action is
good or bad.

[Environment]

The agent must put specified objects into specified containers.
- The hand can hold at most one object.
- Containers have unlimited capacity.
- take: pick up an object; the hand must be empty. An object inside a container
can be taken only when that container is open.
- put: place the held object in an open container.
- drop: place the held object on the table.
- open: open a container; this is also allowed while holding an object.
- An object's goal is complete whenever it is in its specified container,
regardless of whether the container is open or closed. Objects initially in the
correct location also count as complete.
- An object already in its goal location can still be taken away, but doing so
reverses its completed goal.
- The current state lists every object's location. If no object is at the hand
location, the agent's hand is empty.

[Judgment rules]

Judge the four items independently; their answers need not agree. The following
rules define the comparison used here. You are not judging whether actions are
good or bad, or predicting gradients or training outcomes.
History concerns the operations already performed; current state concerns the
remaining tasks and their prerequisites; action concerns the present operation
type; task role concerns what that operation directly changes. Do not
automatically apply one item's comparison criterion to another.

History:
Compare the operation patterns actually experienced and their ordering. Command
sequences need not match word for word.
Object names may map to the same task roles. Distinguish continuing work on a
goal, picking up and returning the same object, and switching to another object
before returning. Judge only from the supplied history and state; do not infer
an unspecified initial state, intention, or outcome.
A complete round trip means that an object leaves a location and returns to it.
Picking it up and placing it elsewhere is a transfer, not a round trip merely
because it includes take and put/drop. Use only location relations that can be
```

established from the records.
First check whether both histories already contain a complete round trip. You may ignore how many times it repeats only if both contain the same kind of complete round trip on the same object. When collapsing repetition counts, retain at least one round.
No round trip on one side versus a round trip on the other is a difference in experience patterns, not an ignorable count difference. For example, put apple in cabinet alone and put apple in cabinet → take apple → put apple in cabinet have dissimilar histories.
Likewise, an added loop of another kind, a switch to another object, or a different task role is not merely another repetition of the original loop.
Preserve operation types in history: drop and put cannot be substituted as the same historical operation just because both free the hand. Names of objects in the same category may be mapped, but differences in operation type cannot be erased.
Histories that only exchange independent operations or complete operation segments may be similar. Independence means that every step remains executable after the exchange, each segment handles the same objects and changes goal progress in the same way, and there are no prerequisite dependencies between segments, such as opening a container or freeing the hand. For a complete segment, also check the held object when entering and leaving it; different objects alone do not establish independence.
For example, given an empty hand and a bottle on the table, the complete round trip take bottle → drop bottle may move before or after a segment handling another object and ending with an empty hand, provided that segment does not depend on the bottle's handling. In contrast, open box cannot move after taking an object from that box. If independence cannot be established, do not assume that operations can be reordered.
If repetition leaves too few steps, account for this in the current-state budget judgment; it does not by itself make the historical patterns different.
An empty history means no preceding actions; an omitted history means missing information.

Current state:
First consider only the task goals and the state before the action, without using either record's chosen action.
Compare the remaining tasks before matching objects by name. Objects may be renamed under the mapping: A's ``only unfinished goal object'' may map to B's differently named ``only unfinished goal object.'' Objects with the same name need not map to each other. The object chosen for the present action does not determine this mapping.
The mapping must remain consistent; do not choose a different object for each condition. Preserve opening dependencies caused by multiple remaining goals sharing a container. Retain the completion status of completed goals, but object and container names unrelated to remaining tasks need not be preserved.
Under this task-role mapping, check each of the following:
- Which goals are complete and which remain unfinished;
- Whether the hand is empty, holds an unfinished goal object, holds a completed goal object, or holds an irrelevant object;
- For every unfinished goal object not already held, whether its current container must be opened before it can be taken;
- Whether its destination container is open.
Judge states similar only when these key conditions correspond. Merely having ``two goals complete'' or ``both need at least three steps to finish'' is insufficient. Conversely, if each side has only one unfinished, directly accessible goal object, with corresponding held-object conditions, an open destination, and the same budget conditions, do not judge them dissimilar solely because one object is a cup and the other a bottle. Locations establish goal progress, holding, and access prerequisites; location names or storage forms need not themselves match.
In this environment, neither an unfinished object on the table nor one in an open container requires opening a container first. Their states can be similar when the other key conditions correspond. An object in a closed container still requires opening it.

```
Object names may differ, but missing prerequisites cannot be exchanged.
In particular, needing to free the hand and needing to open a container are
different missing conditions. Judge the states dissimilar even if both require
one preparatory step and the same minimum number of steps to finish.
Differences that leave these conditions unchanged can be ignored, such as an
irrelevant object moving between two non-hand locations, or a container
unrelated to remaining goals changing its open/closed state. An irrelevant
object occupying the hand is not an ignorable difference.
Completed goal objects still count toward goal progress. However, if their
container is unrelated to all remaining goals, changing only whether it is open
does not affect state similarity.
Different histories or present choices do not directly determine current-state
similarity.
A difference in remaining steps counts against state similarity only if it
changes the feasibility of finishing the remaining goals or the choices that are
necessary.

Action:
Compare the operation itself.
Identical commands are similar actions.
Changing the object or destination container within the same operation usually
still gives similar actions.
take, put, drop, and open are different operations.
Leave differences in task purpose to the ``action's task role'' judgment.

The action's task role:
Use the state and task at execution time to judge what the action actually does.
Examples include acquiring an unfinished goal object, removing a completed goal
object, completing a goal placement, placing an object elsewhere, releasing an
irrelevant object to free the hand, and opening a container needed to acquire or
place a goal object.
Compare direct outcomes: how goal completion changes, how holding changes, and
how the prerequisites for taking or placing goal objects change. If an action
changes several of these, compare them together rather than selecting only one
effect.
Thus, releasing an irrelevant object onto the table or into an open container
has a similar task role if neither changes goal progress or necessary
prerequisites. Containers have unlimited capacity here; do not invent an
obstruction to later placements due to occupied space.
For a held, unfinished goal object, placing it on the table and placing it in an
open non-goal container also have similar task roles if both leave its goal
unfinished, both free the hand, and the object remains directly retrievable. Do
not infer intentional temporary storage on one side and an error on the other.
Placing it in the goal container also completes its goal, so ``both free the
hand'' is not enough to judge the roles similar.
Identical commands may have different task roles, and different commands may
have similar roles. Compare direct roles, without guessing whether a future
strategy is worthwhile. Different remaining step counts alone do not make the
roles different; the budget's effect on overall feasibility belongs in the
current-state comparison.
Do not judge task roles similar solely because both actions appear reasonable.

Judge only from the supplied information; do not invent states, histories, or
subsequent outcomes.
Each answer must be ``similar'', ``dissimilar'', or ``insufficient
information''.
Use ``insufficient information'' only when information necessary for the
judgment is missing.

[Examples]

In the examples below, all unmentioned conditions are the same and the remaining
step budget is sufficient.

Example 1: Different names, similar experience and decisions
```

```
Record A:
Task: put the book in the cabinet.
History: open cabinet.
Current state: empty hand, book on the table, cabinet open.
Action: take book.

Record B:
Task: put the cup in the basket.
History: open basket.
Current state: empty hand, cup on the table, basket open.
Action: take cup.

Output:
{"history":"similar","current state":"similar","action":"similar","task role of
action":"similar"}

Example 2: Identical current states, different choices

Record A:
Task: put the book in the cabinet.
History: take book.
Current state: holding the book, cabinet open.
Action: put book in cabinet.

Record B:
Task: put the book in the cabinet.
History: take book.
Current state: holding the book, cabinet open.
Action: drop book.

Output:
{"history":"similar","current state":"similar","action":"dissimilar","task role
of action":"dissimilar"}

Example 3: Identical commands, different task roles

Record A:
Task: move the cup from the box to the basket.
History: open box.
Current state: empty hand, cup in the open box, cup goal unfinished.
Action: take cup.

Record B:
Task: keep the cup in the box and put the book in the cabinet.
History: open box.
Current state: empty hand, cup in the open box, cup goal complete, book goal
unfinished.
Action: take cup.

Output:
{"history":"similar","current state":"dissimilar","action":"similar","task role
of action":"dissimilar"}

Example 4: Different histories, identical current states

Both records have the task of putting the book in the cabinet. The key and pen
are not goal objects and are initially on the table.

Record A:
History: take key → drop key → take pen → drop pen.
Current state: empty hand, book, key, and pen on the table, cabinet open.
Action: take book.

Record B:
History: take book → drop book → take book → drop book.
Current state: empty hand, book, key, and pen on the table, cabinet open.
Action: take book.

Output:
{"history":"dissimilar","current state":"similar","action":"similar","task role
of action":"similar"}
```

```
Example 5: Different operations, both releasing an irrelevant object

Both records have the task of putting the book in the cabinet. The key is not a
goal object.

Record A:
History: empty.
Current state: holding the key, book on the table, box and cabinet open.
Action: drop key.

Record B:
History: empty.
Current state: holding the key, book on the table, box and cabinet open.
Action: put key in box.

Output:
{"history":"similar","current state":"similar","action":"dissimilar","task role
of action":"similar"}

Example 6: Both place an object, but the destination changes the task role

Record A:
Task: put the book in the cabinet.
History: take book.
Current state: holding the book, cabinet and basket open.
Action: put book in cabinet.

Record B:
Task: put the book in the cabinet.
History: take book.
Current state: holding the book, cabinet and basket open.
Action: put book in basket.

Output:
{"history":"similar","current state":"similar","action":"similar","task role of
action":"dissimilar"}

Example 7: Both need one preparatory step, but different prerequisites are
missing

Both records have the task of putting the book in the cabinet; the key is not a
goal object.

Record A:
History: empty.
Current state: holding the key, book in the open box, cabinet open.
Action: drop key.

Record B:
History: empty.
Current state: empty hand, key on the table, book in the closed box, cabinet
open.
Action: open box.

A must first free the hand, while B must first open the box. Do not judge the
states similar by reducing both to ``needing preparation''; freeing the hand and
opening access also have different task roles.
Output:
{"history":"similar","current state":"dissimilar","action":"dissimilar","task
role of action":"dissimilar"}

Example 8: Additional available actions differ, but conditions for the remaining
goal are the same

Both records have the task of putting the book in the cabinet and the pen in the
basket.

Record A:
History: empty.
Current state: empty hand, book in the closed box, cabinet open; pen already in
```

```
the basket, basket closed.
Action: open box.

Record B:
History: empty.
Current state: empty hand, book in the closed box, cabinet open; pen already in
the basket, basket open.
Action: open box.

The pen goal is complete on both sides, and both require opening the box to
access the book. Whether the basket is open changes only additional available
operations, not the conditions for completing the remaining goal.
Output:
{"history":"similar","current state":"similar","action":"similar","task role of
action":"similar"}

Example 9: Different storage forms, identical access prerequisites

Both records have the task of putting the book in the cabinet; both have an
empty hand and an open cabinet.
Record A: empty history; book on the table; current action take book.
Record B: empty history; book in the open box; current action take book.
Both can take the book directly, without opening a container first.
Output:
{"history":"similar","current state":"similar","action":"similar","task role of
action":"similar"}

Example 10: Different repetition counts of the same loop

Both records have the same task: put the book in the cabinet. The book is
initially on the table and the cabinet is open.
Record A: history take book → drop book; currently empty-handed with the book on
the table; current action take book.
Record B: history take book → drop book → take book → drop book; currently
empty-handed with the book on the table; current action take book.
Both histories pick up the same goal object and return it to the table; B merely
repeats the complete loop once more.
Output:
{"history":"similar","current state":"similar","action":"similar","task role of
action":"similar"}

[Output requirements]

Output only one JSON object with the following four keys:
``history'', ``current state'', ``action'', and ``task role of action''.

Do not output explanations, extra fields, or Markdown markers.
```

G.2 Full Seven-Field Relation Prompt for the First Update

This is the version actually used for the detailed same-sign audit in the first update. We report five fields: alternative choices, operation type, direct effect, task role, and task progress. The other two fields were present in the original inference requests and have not been retrospectively removed from the prompt below. *Translated from the original prompt.*

First-update seven-field relation audit

```
Compare the task relations between two independent action records. Use only the
supplied facts. Do not judge whether actions are good or bad, explain gradients,
credit, or probabilities, or infer intentions. Relations may overlap, or none
may hold.

[Environment and facts]
The task is to put specified objects in specified containers. The hand can hold
only one object, and containers have unlimited capacity.
take requires an empty hand; taking an object from a container also requires
that container to be open. put places the held object in an open container. drop
```

places the held object on the table. open opens a closed container and can also be executed while holding an object.
A goal is complete when its object is in the specified container; taking it away reverses completion. Irrelevant objects have no specified destination.
The input provides the complete current state, legal commands, and changes made by the actual action as computed by the environment. These are facts and need not be restated. The records are not consecutive; do not move one record's state into the other.

[Comparison criteria]
Differently named objects may correspond by task role; identically named objects may have different roles. Object roles are: unfinished goal object, completed goal object, and irrelevant object.
The correspondence must preserve object relations relevant to the actions being compared: one object and its source/destination containers cannot use separate mappings within a single correspondence. Different objects cannot be merged.
Unrelated global goal counts need not match.
Judge command form, direct physical effect, and task role separately. The same task role does not require the same command or exact destination and does not imply the same long-term value.
Only ``alternative choices'' requires both choices to be executable in both original states. Prerequisite--execution and complementary prerequisites compare operation dependencies; they do not require identical states, both states to lack both prerequisites, or each action to be executable in the other record's original state.
Judge only the operation structure in the current task, not optimality, identical histories, or whether time is sufficient.

[Item-by-item judgments]
1. Alternative choices
Keep the actual actions of A and B fixed. Can another legal command be found on each side to form two corresponding choices?
The input's ``candidates'' already lists all cross-combinations that match command types and differ from the actual action. Check them one by one; do not invent commands or replace actual actions.
A candidate holds only when A's other command maps to B's actual action and B's other command maps to A's actual action, each column preserves operation, object role, and task role, the two commands on each side differ, and the object-relation mapping above is consistent.
For example, when holding an unfinished goal object with its destination open, correct placement and dropping can correspond. When holding an irrelevant object, placing it in an open container and dropping can correspond: both free the hand, but are still different commands. When mapping taking an object and opening that object's destination, do not substitute taking another object with a different destination.
If the relation holds, provide one passing candidate number. A negative judgment requires that every candidate fails; do not reject a candidate just because literal names differ.

2. Same operation type
Compare only the take/put/drop/open type.

3. Same direct effect
The actual changes share at least one type: bringing an object into the hand, freeing the hand, opening a container, completing a goal, or reversing a goal. Correct placement and dropping both free the hand, but retain the distinction that the former completes a goal.

4. Prerequisite--execution
Under a task-role mapping, does a condition newly supplied by one actual action serve as an execution prerequisite for the other actual operation? Both actual actions must be used.
Examples: opening a goal object's source container → taking the goal object; opening the destination container → placing the goal object; freeing the hand → taking another unfinished object.
It is normal for the executing action's prerequisites to be already satisfied;

```
that does not negate the structural relation. The records need not be
consecutive. Conversely, merely saying that one action prepares for an imagined
third action does not establish this relation.
The dependency must serve a task already unfinished before execution. Do not
treat removing a completed object and putting it back, or dropping an object and
taking it back, as task cooperation through a recovery loop. Opening a container
→ placing the held object inside to free the hand can qualify if freeing the
hand serves another unfinished object; this preparation need not be the only
possible way.
If the relation holds, state the direction and prerequisite, and explain how the
other actual action uses it. Across tasks, state the corresponding
object/container roles.

5. Complementary prerequisites
Do the two actual actions supply two different types of prerequisite for the
same task operation under a role mapping? First state the operation served by
each action, then map roles; the two records need not serve the same literal
command. State the operation and each prerequisite.
Acquiring an unfinished object + opening its destination respectively provide
possession of the object and an open container, preparing for put. For example,
A takes the book to supply possession for put book in cabinet, while B opens the
basket to supply an open container for put bottle in basket. The relation holds
under book↔bottle and cabinet↔basket, even if A's cabinet is already open.
Freeing the hand + opening an unfinished object's source container respectively
provide an empty hand and open access, preparing for take.
Two take actions both supply possession; two set-down actions both supply an
empty hand. Different objects do not make these complementary. Freeing the hand
and opening a destination container cannot merge take and put into one step.
Merely serving the same final goal is insufficient.

6. Same task role
Compare the manipulated object's role, goal completion/reversal, changes in
holding, and whether opening a container provides access for taking or placing
in the current task. Preserve these direct roles; do not additionally compare
which object might be taken later or how many goals could be served.
For example, acquiring an unfinished goal object differs from acquiring a
completed one; correct placement differs from placing in a non-goal location.
Putting an unfinished object in a non-goal container and dropping it on the
table both release the goal object without completing its goal; do not judge
them different solely because of put/drop or the destination. Two ways of
setting down an irrelevant object to free the hand can also match. Judge freeing
the hand by the current change in holding, not by different numbers of remaining
goals.
Opening a source container for take and opening a destination container for put
have different task roles. When opening one container serves several purposes,
do not omit any of its task-prerequisite roles.

7. Opposite task progress
This holds only when one action completes a goal and the other reverses an
already completed goal with a corresponding role. Taking an unfinished object
does not complete a goal; dropping a held object does not automatically reverse
an already completed goal.

[Output]
Output JSON only. Judgments are ``holds'', ``does not hold'', ``insufficient
information'', or ``judgment incomplete''. Use insufficient information only
when necessary input facts are missing. If the check is unfinished, use judgment
incomplete rather than presenting it as does not hold. The rationale should
usually be one sentence citing concrete facts. Give only relation evidence,
without restating the full state. Retain all fields and use an empty string for
inapplicable evidence fields.
{
"alternative choice": {"judgment":"", "candidate id":"", "rationale":""},
"same operation type": {"judgment":"", "rationale":""},
"shared direct effect": {"judgment":"", "rationale":""},
```

```
"prerequisite--execution": {"judgment":"", "direction":"", "prerequisite":"",
"rationale":""},
"complementary prerequisites": {"judgment":"", "common operation":"", "A
condition":"", "B condition":"", "rationale":""},
"same task role": {"judgment":"", "rationale":""},
"opposite task progress": {"judgment":"", "rationale":""}
}
When prerequisite--execution holds, its direction must be ``A to B'', ``B to
A'', or ``bidirectional''. For complementary prerequisites, set the common
operation to take or put, and explain the corresponding objects and tasks in the
rationale. Do not use unescaped English double quotes inside strings.
```

G.3 Full Five-Field Relation Prompt for Updates 2–5

The later requests use the following version. It removes prerequisite–execution, complementary prerequisites, and their explanations from the preceding version; the remaining relations must still be judged in each record's own state. We list both actual versions to make the difference between the first-update consensus audit and the later original-order audit explicit, not to interpret a prompt-version change as a training-stage effect. *Translated from the original prompt.*

Updates 2–5 five-field relation audit

```
Compare the task relations between two independent action records. Use only the
supplied facts. Do not judge whether actions are good or bad, explain gradients,
credit, or probabilities, or infer intentions. Relations may overlap, or none
may hold.

[Environment and facts]
The task is to put specified objects in specified containers. The hand can hold
only one object, and containers have unlimited capacity.
take requires an empty hand; taking an object from a container also requires
that container to be open. put places the held object in an open container. drop
places the held object on the table. open opens a closed container and can also
be executed while holding an object.
A goal is complete when its object is in the specified container; taking it away
reverses completion. Irrelevant objects have no specified destination.
The input provides the complete current state, legal commands, and changes made
by the actual action as computed by the environment. These are facts and need
not be restated. The records are not consecutive; do not move one record's state
into the other.

[Comparison criteria]
Differently named objects may correspond by task role; identically named objects
may have different roles. Object roles are: unfinished goal object, completed
goal object, and irrelevant object.
The correspondence must preserve object relations relevant to the actions being
compared: one object and its source/destination containers cannot use separate
mappings within a single correspondence. Different objects cannot be merged.
Unrelated global goal counts need not match.
Judge command form, direct physical effect, and task role separately. The same
task role does not require the same command or exact destination and does not
imply the same long-term value.
Only ``alternative choices'' requires both choices to be executable in both
original states.
Judge only the operation structure in the current task, not optimality,
identical histories, or whether time is sufficient.

[Item-by-item judgments]
1. Alternative choices
Keep the actual actions of A and B fixed. Can another legal command be found on
each side to form two corresponding choices?
The input's ``candidates'' already lists all cross-combinations that match
command types and differ from the actual action. Check them one by one; do not
```

```
invent commands or replace actual actions.
A candidate holds only when A's other command maps to B's actual action and B's
other command maps to A's actual action, each column preserves operation, object
role, and task role, the two commands on each side differ, and the
object-relation mapping above is consistent.
For example, when holding an unfinished goal object with its destination open,
correct placement and dropping can correspond. When holding an irrelevant
object, placing it in an open container and dropping can correspond: both free
the hand, but are still different commands. When mapping taking an object and
opening that object's destination, do not substitute taking another object with
a different destination.
If the relation holds, provide one passing candidate number. A negative judgment
requires that every candidate fails; do not reject a candidate just because
literal names differ.

2. Same operation type
Compare only the take/put/drop/open type.

3. Same direct effect
The actual changes share at least one type: bringing an object into the hand,
freeing the hand, opening a container, completing a goal, or reversing a goal.
Correct placement and dropping both free the hand, but retain the distinction
that the former completes a goal.

4. Same task role
Compare the manipulated object's role, goal completion/reversal, changes in
holding, and whether opening a container provides access for taking or placing
in the current task. Preserve these direct roles; do not additionally compare
which object might be taken later or how many goals could be served.
For example, acquiring an unfinished goal object differs from acquiring a
completed one; correct placement differs from placing in a non-goal location.
Putting an unfinished object in a non-goal container and dropping it on the
table both release the goal object without completing its goal; do not judge
them different solely because of put/drop or the destination. Two ways of
setting down an irrelevant object to free the hand can also match. Judge freeing
the hand by the current change in holding, not by different numbers of remaining
goals.
Opening a source container for take and opening a destination container for put
have different task roles. When opening one container serves several purposes,
do not omit any of its task-prerequisite roles.

5. Opposite task progress
This holds only when one action completes a goal and the other reverses an
already completed goal with a corresponding role. Taking an unfinished object
does not complete a goal; dropping a held object does not automatically reverse
an already completed goal.

[Output]
Output JSON only. Judgments are ``holds'', ``does not hold'', ``insufficient
information'', or ``judgment incomplete''. Use insufficient information only
when necessary input facts are missing. If the check is unfinished, use judgment
incomplete rather than presenting it as does not hold. The rationale should
usually be one sentence citing concrete facts. Give only relation evidence,
without restating the full state. Retain all fields and use an empty string for
inapplicable evidence fields.
{
"alternative choice": {"judgment":"", "candidate id":"", "rationale":""},
"same operation type": {"judgment":"", "rationale":""},
"shared direct effect": {"judgment":"", "rationale":""},
"same task role": {"judgment":"", "rationale":""},
"opposite task progress": {"judgment":"", "rationale":""}
}
```