FROM LOCAL IMPASSES TO GROUNDED RECOVERY: EXPERIENCE REUSE FOR LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM agents can exhaust their interaction budgets by repeating unproductive actions, even when relevant past experience is available. Effective experience reuse must therefore do more than retrieve a similar task: it must identify a supported way to restore progress under current conditions. We propose Policy–Memory–Recovery (PMR), a framework for recovering from local impasses without updating policy parameters. PMR builds on a key observation: the experience revealing a difficulty need not contain its solution, and evidence for a useful local effect can occur in an otherwise unsuccessful episode. PMR separately establishes context and effect evidence, then links compatible evidence across episodes into conditional recovery records. This allows unsuccessful interaction to supply useful evidence without treating its behavior as a solution. At deployment, retrieved experience advises the policy, while corrective execution requires current-state checks and observation-based confirmation of progress, without auxiliary recovery-generation calls. Across multiple long-horizon interactive environments, PMR consistently achieves the highest mean task success, outperforming strong baselines by 3.3–22.6 percentage points while using fewer test-time actions and model calls. Controlled support removals and a fixed-bank execution intervention show that the gain follows relevant evidence and checked execution rather than memory volume alone, establishing evidence-grounded recovery as a more effective alternative to task-level experience retrieval.[1]

1 INTRODUCTION

Large language model (LLM) agents can reason and act (Yao et al., 2023b) yet still make poor sequential decisions (Liu et al., 2024). Repeatedly searching the same locations, retrying actions with unmet prerequisites, or revisiting unchanged states consumes a finite interaction budget while leaving the obstacle unresolved. We call such an interval an *observable local impasse*, or *trap*: a required part of the task remains unfinished and recurrent behavior yields no goal-relevant progress. Progress can mean satisfying a subgoal, acquiring a prerequisite, or obtaining information needed to continue. An impasse is therefore more specific than a long episode and need not make the task unsolvable. The challenge is to turn continued interaction into a change that addresses the obstacle.

Past experience offers a way to improve decisions without updating policy weights (Zhao et al., 2024), yet retrieved experience can propagate errors or induce misaligned replay (Xiong et al., 2025). Reflexion turns feedback into verbal reflections (Shinn et al., 2023); Synapse retrieves abstracted trajectory exemplars (Zheng et al., 2024); AWM induces reusable workflows that guide generation (Wang et al., 2025). Procedural reuse also includes executable skill libraries (Wang et al., 2024), but relevance to the overall task does not establish relevance to its current obstacle. Memory representation and retrieval organization shape transfer (Hu et al., 2026); LifeMem distills skills from workflow clusters (Qiu et al., 2026). PMR focuses on what local evidence supports a recovery and what current conditions authorize it.

Even a candidate that addresses the right obstacle raises a second question: *what does the experience establish about its expected effect?* An episode is one complete task attempt; its final outcome does not label every transition inside it. A failed episode can contain a useful state change, while a

[1] Code for reproducing the main experiments is provided in the supplementary materials.

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.

successful episode can contain ineffective actions. ReasoningBank already distills strategies from successful and failed experience (Ouyang et al., 2026). The remaining issue is not merely admitting failures, but identifying the observations that support a particular recovery effect. Acquiring an object or revealing its location provides local effect evidence; attempts where the intended change does not occur provide *non-effect evidence* that constrains support for the candidate. A general lesson and an empirically supported local effect answer different questions.

Crucially, evidence of an obstacle and evidence of its resolution can occur in different episodes. One episode may repeatedly fail to locate a mug, while another records that opening a cabinet reveals an object of the needed type. The first establishes an unresolved context; the second supports an information-producing action. A useful association links them through the unmet requirement and compatible conditions, rather than requiring one trajectory to demonstrate both the trap and its escape. The sources must remain distinct in the record: observing a difficulty does not demonstrate that a proposed remedy works. This organization lets unsuccessful interaction contribute context, non-effect evidence, or a locally effective transition without treating its entire behavior as a solution. The resulting reusable unit is a *conditional recovery opportunity*: a supported candidate for addressing a particular local obstacle.

Historical support still does not determine whether that candidate is executable now. AdaMEM adapts to evolving test-time conditions by retrieving long-term experience and synthesizing short-term strategies (Zhang et al., 2026); its Step-MFT further targets strategies that change decisions and promote success. This advances both the responsiveness and utility of guidance. Our question is complementary: how should a supported candidate become an authorized intervention? In the search example, a historical cabinet location is not a current location estimate. The action must be *grounded* (Ichter et al., 2023) by binding its references to valid entities in the current observed state, satisfy prerequisites, obey action rules, and fit the remaining budget. An *admission check* decides whether it can be attempted; subsequent observations establish whether its intended local effect occurred. Updating advice and checking execution are distinct responsibilities.

These distinctions also clarify the relationship to guarded workflows and trap-aware learning. ReUseIt synthesizes workflows with execution guards from successful and failed attempts (Liu et al., 2026), and Soar provides a classical precedent for impasse-driven procedural learning (Laird et al., 1986). T3 uses observable indicators of belief trapping to truncate uninformative training trajectories and improve reinforcement-learning credit assignment (Zou et al., 2026). We instead study experience-based recovery during deployment with fixed policy parameters. Our contribution connects separately established context and effect evidence to current-state execution checks, rather than introducing failure reuse or guards alone. This leads to the central question: *how can past interaction provide locally supported recovery options, and what should authorize an LLM agent to execute them at a new impasse?*

To address these gaps, we propose **Policy–Memory–Recovery (PMR)**, which converts local interaction evidence into grounded recovery while keeping the policy fixed. Its conditional records separate evidence of an unresolved context from evidence of a useful effect, while its recovery controller separates policy advice from action authority and checks candidate execution against the current state. Together, these choices address the two limitations above: task-level relevance alone does not establish a locally useful effect, and historical experience alone does not establish that an action is valid now. Across long-horizon interactive tasks, PMR improves task success while reducing online actions and model calls, and controlled interventions show that these gains follow relevant support and checked execution. Figure 1 summarizes the resulting observe–compile–ground pipeline.

Contributions.

1. **Outcome-decoupled recovery evidence.** Conditional records make local contexts, supported effects, non-effects, and provenance reusable across episodes, allowing unsuccessful interaction to expand recovery coverage without replaying entire failed trajectories (Sections 3.1–3.2).
2. **Condition-checked recovery authority.** Separating policy advice from corrective execution turns retrieved experience into state-grounded interventions that are admitted only when their support and current conditions permit execution (Sections 3.3–3.4).
3. **Evidence-to-utility analysis.** Retention laws and controlled interventions connect evidence relevance and source diversity to recovery coverage, task success, and online efficiency.

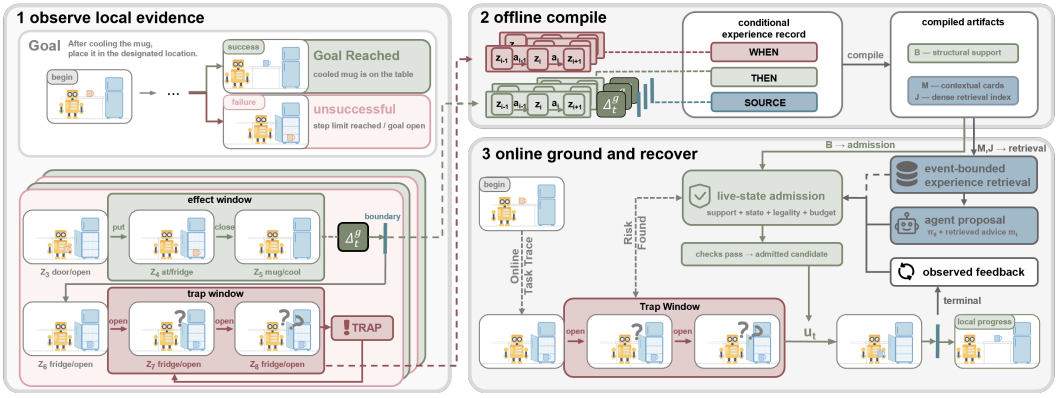

Figure 1: **PMR: from local evidence to grounded recovery. (1) Observe.** The mug example separates an effect window, where an observed change Δ_t^g advances a local obligation, from a trap window, where repeated actions leave the goal open. **(2) Compile.** Compatible context and effect traces are linked with their conditions and provenance in conditional records, producing structural support $\mathcal{B}$, event-bounded cards $\mathcal{M}$, and a dense retrieval index $\mathcal{J}$. **(3) Ground and recover.** Retrieved cards advise the frozen policy; corrective candidates are admitted only after current-state checks, and observed feedback confirms local progress or releases the active stage.

2 Problem Formulation

Interaction and historical experience. Under partial observability (Kaelbling et al., 1998), an agent with frozen policy π_θ pursues goal g through actions a_t and observations o_t, with history $h_t = (o_0, a_0, \ldots, a_{t-1}, o_t)$. An episode ends at environment termination or after B submitted actions, with terminal success $y \in \{0, 1\}$. We assume an observable interface

$$z_t = \Phi(g, h_t) = (F_t, U_t, \mathcal{A}_t), \tag{1}$$

where F_t contains observation-supported facts, U_t outstanding goal obligations and remaining counts, and $\mathcal{A}_t$ the available action interface. The projection reads the goal and received observations, without hidden state or future outcomes. Let Δ_t^g denote goal-relevant progress observed after a_t, with $\Delta_t^g = \varnothing$ indicating no observed progress. Historical experience is a fixed pool $\mathcal{D} = \{(\tau_i, \sigma_i, y_i)\}_{i=1}^N$, comprising traces, source-task identities, and terminal outcomes. It may contain both successful and unsuccessful episodes.

Local evidence reuse for recovery. Given $\mathcal{D}$, π_θ, and Φ, we seek experience-based interventions that address local obstacles and improve terminal task success under budget B. Policy parameters remain fixed, intervention decisions use only available observations, and corrective actions count against the same budget as policy actions. Section 3.1 operationalizes the observable impasses introduced in Section 1. Following fine-grained agent evaluation (Ma et al., 2024), we distinguish local progress from task completion; evaluation measures terminal success together with action and inference costs.

3 Method

PMR implements the three stages in Figure 1: **observe** local effect and impasse windows, **compile** compatible evidence into conditional records offline, and **ground and recover** under current observations online. The record's WHEN, THEN, and SOURCE fields summarize applicability, supported effects, and provenance; Section 3.2 formalizes their content. Following modular agent architectures (Sumers et al., 2024), retrieved cards advise the frozen policy, whereas recovery uses structural support and live-state admission to decide whether a correction may replace its proposal. Environment feedback then advances a local stage, releases the branch, or returns control to the policy. Let $\mathcal{B}$, $\mathcal{M}$, and $\mathcal{J}$ denote structural support, contextual cards, and their retrieval index, compiled from $\mathcal{D}$ under configuration λ. These artifacts, λ, and policy parameters θ remain fixed during evaluation; only history and retrieval/controller states evolve. Structural support supplies execution

evidence, while cards render bounded experience as policy advice. Complete algorithms and adapter constructions are in Appendices A and C.

Experience-induced recovery under a shared interface. The adapter defines an observable interface for goal obligations, action grounding, and progress verification, while offline source trajectories determine the recovery content compiled within it. Observed transitions provide context–effect associations, supporting and non-effect evidence, and executable prefixes used to retain conditional recovery candidates. At deployment, PMR grounds these candidates in the current observation and admits them only when their prerequisites hold. All three environments instantiate this shared evidence-compilation and live-state-admission principle through their native observation and action interfaces; implementation details appear in Appendix C.

3.1 Goal-relative impasse discovery

Using z_t and Δ_t^g from Section 2, discovery identifies intervals W with $U_t \neq \varnothing$ and $\Delta_t^g = \varnothing$ for all $t \in W$, together with a configured stagnation predicate. This predicate detects abstract-state recurrence, repeated actions without an observed event, or repeated inadmissible actions. Progress includes satisfying an obligation, acquiring a prerequisite, revealing relevant information, or restoring an adapter-exposed affordance. Detection does not use terminal success, so unsuccessful episodes can supply impasse evidence. Each impasse record $i \in \mathcal{I}$ stores its entry context κ_i and a source reference into $\mathcal{D}$. The projection retains local conditions such as pending object types, resource occupancy, and direct affordances rather than the full task description. Matching compares these represented conditions across instances, not hidden states. Adapter-specific predicates and extraction rules are in Appendix C.

3.2 Linking impasses to recovery evidence

A discovered impasse identifies a need, not a solution. A *conditional experience record* represents the conditional recovery opportunity introduced in Section 1: a context linked to separately supported effects. Equation 2 is the record schema; Algorithm 1 constructs these fields and Algorithm 2 consumes them during interaction. Reading the context and its effect references together yields

$$c = (\kappa_c, P_c, \mathbf{e}_c, \mathbf{T}_c, \mathbf{b}_c; \mathcal{S}_c, \mathcal{V}_c), \tag{2}$$

where κ_c is the context, P_c the applicability conditions, $\mathbf{e}_c$ the effect sequence, $\mathbf{T}_c$ observable terminals, and $\mathbf{b}_c$ positive action budgets. Here an observable terminal is a local stage-completion condition within an episode. The source set $\mathcal{S}_c$ identifies context evidence, whereas $\mathcal{V}_c$ records empirical support for the referenced effects, including observed non-effects. Keeping them separate lets a failed episode supply a stalled context while another episode supplies the effect evidence, forming a cross-episode recovery candidate grounded online. Concretely, the mug example reads as: *WHEN* the mug-finding obligation remains unresolved and the current observation grounds a legal cabinet action; *THEN* attempt the supported opening effect and wait for the observable terminal that confirms the needed information; *SOURCE* keep the stalled context episode in $\mathcal{S}_c$ and the effect and non-effect observations in $\mathcal{V}_c$. The historical cabinet location remains evidence for the candidate, not a current-state fact. Compilation links compatible conditions and effects, not episode outcomes.

Inducing supported effects. As in action-model learning (Yang et al., 2007), effects are induced from experience: here they are changes in the observable projection or endpoints of bounded continuations. Their representation follows the interface: where typed arguments are available, normalization retains relevant types while replacing instance-specific arguments with roles. Candidate support is assessed from the source transitions and continuations that exhibit the effect. An eligible legal attempt without that effect provides *non-effect* evidence, rather than inheriting a negative label from the episode outcome. The accepted candidate set depends on fixed, branch-specific criteria; the concrete partitions, counts, and thresholds are specified in Appendix C. This separation prevents a final failure from automatically invalidating an earlier useful transition.

Composing and associating recoveries. Compilation associates each context with a supported effect or bounded continuation that addresses its unresolved obligation under compatible projected conditions. Following the precondition–effect representation of classical planning (Fikes & Nilsson, 1971), explicit prerequisites and effects permit composition: an observed effect of one stage

supports a prerequisite of the next. Each stage retains its prerequisite, observable terminal, budget, and sources, giving an option-like control structure (Sutton et al., 1999). Interfaces with executable local sequences can instead supply supported continuations directly. Live-state admission checks the remaining prerequisites before execution. Direct-effect and information support are constructed separately. In ALFWorld, one selected chain supplies context-linked suffixes; direct-effect and information branches are separate. Record and context-class counts therefore quantify context-source diversity over the linked sequences (Appendix C). The impasse identifies what is missing; effect evidence identifies a supported change. Compatibility links these through represented conditions rather than whole-task similarity. In the mug example, the link supports information acquisition; current observations determine where the target is found. Both source links make the composed recovery candidate inspectable.

Unsuccessful episodes can therefore contribute contexts, non-effect evidence, or locally valid transitions. Section 4.4 traces this provenance into retained support and tests its local recovery utility. The separately admitted direct-effect branch complements context-linked chains when a supported goal effect is already applicable.

3.3 Event-bounded experience retrieval

As in retrieval-augmented generation (Lewis et al., 2020), textual cards advise the policy through short source segments ending at an observed progress event, with role-bound action examples for in-context learning (Rubin et al., 2022) and links to supporting records or effects. Eligibility is fixed by the adapter separately from structural discovery; in ALFWorld, cards require successful source episodes and associated chain records (Appendix C). Unsuccessful sources contribute structural evidence, while advice cards retain their successful-source and structural-link requirements.

When the retrieval gate permits, cards are filtered by outstanding obligations and controller context, ranked by dense similarity (Karpukhin et al., 2020) through $\mathcal{J}$, and limited to k distinct event signatures. Rendering binds roles into temporary context m_t; historical locations remain examples, not current-state facts. Otherwise $m_t = \varnothing$. The frozen policy proposes $\widetilde{a}_t \sim \pi_\theta(\cdot \mid g, h_t, m_t)$. Scheduling and filtering details appear in Algorithm 2. Retrieved cards advise the policy, while recovery can consult structural support outside the retrieved top-k. Retrieval identifies potentially useful experience; admission checks whether a proposed correction is supported and executable now. This separates advisory relevance from execution authority.

3.4 Condition-checked recovery

The recovery channel grounds supported candidates before allowing them to replace a policy action. ALFWorld implements three branches: **chain recovery** follows a supported effect sequence; **direct-effect recovery** addresses an outstanding obligation; and **information recovery** seeks information after a repeated action without an observed event. Information recovery requires live stagnation, whereas chain and direct-effect recovery may act before a new stall when their conditions hold.

Let r_t track the active branch/stage, bindings, budgets, and retry history. The pre-action operator $\mathrm{Resolve}_\Phi(\mathcal{B}, z_t, r_t, \widetilde{a}_t)$ returns proposal u_t, admission bit γ_t, and updated state $\bar{r}_t$. It applies fixed branch arbitration and the relevant support, context, prerequisite, preservation, legality, and budget checks; absent or rejected proposals have $\gamma_t = 0$. The submitted action is

$$a_t = \begin{cases} \mathrm{Action}(u_t), & \gamma_t = 1, \\ \widetilde{a}_t, & \text{otherwise.} \end{cases} \tag{3}$$

After execution, the new observation updates history and controller state. A stage advances only when its terminal is observed. Failed prerequisites, unavailable grounding, exhausted budgets, or applicable stagnation limits release control; retry history constrains renewed intervention under the same recorded condition. Algorithm 2 gives the state updates, and Appendix C specifies branch checks and arbitration. All submitted actions share the episode budget; policy calls are counted even when their proposals are replaced. Recovery uses compiled support without auxiliary online recovery-generation calls. The attempt–observe–continue-or-release loop grounds recovery in current feedback. Observed terminals confirm stage completion; further attempts remain subject to checks and limits. Release restores policy fallback unless another candidate is admitted.

3.5 FROM SOURCE COVERAGE TO RECOVERY UTILITY

Algorithms 1–2 define the operational pipeline; the following identities analyze how compiled support translates into coverage and recovery utility. Compilation preserves source links so that coverage and behavioral utility can be measured separately. For a fixed compiled bank, effect catalog, and matching rule, source retention gives the following exact coverage probability. A source episode is retained as one unit, even if it supplies multiple records.

Proposition 1 (Source retention and coverage)**.** *Partition the source episodes into strata $h = 1, \dots, H$, with N_h episodes per stratum. For a fixed checkpoint x, let $d_h(x)$ count distinct source episodes with matching records in the full fixed bank, before retention. Select n_h episodes uniformly without replacement in each stratum, independently across strata conditional on the quotas. If coverage $s_{\mathbf{n}}(x)$ means retaining at least one such record, then*

$$\Pr\{s_{\mathbf{n}}(x) = 1\} = 1 - \prod_{h=1}^{H} \frac{\binom{N_h - d_h(x)}{n_h}}{\binom{N_h}{n_h}}. \tag{4}$$

Here $0 \le n_h \le N_h$ and $\binom{a}{b} = 0$ when $b > a$. The product is the probability that every stratum misses all $d_h(x)$ supporting episodes, so its complement is the probability that checkpoint x retains at least one witness. Expected coverage of a checkpoint set is the sum of these probabilities, without requiring independence between checkpoints; Figure 4 tests this support-availability prediction.

The proof is in Appendix E.4. The identity distinguishes unique support from redundancy: a sole supporting episode enables coverage, whereas additional supporting episodes protect it against removal. Outcome-stratified retention exposes what is lost when unsuccessful sources are excluded.

From support availability to behavioral utility. Coverage predicts support availability; paired continuations measure its behavioral value. Equation 17 separates gains on newly covered checkpoints from changes at unchanged coverage. Figure 3 instantiates this decomposition, while Figure 4 tests the retention law. The survival identity (Equation 13) links recovery timing to actions saved. Together, these analyses connect source evidence to recovery outcomes and efficiency; Appendix C.3 proves the observation-witness invariant (Invariant 1) and gives execution accounting.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Environments and task selection. We evaluate household search and manipulation in ALFWorld (Shridhar et al., 2021), recipe- and inventory-dependent crafting in TextCraft (Prasad et al., 2024), and language-guided navigation and manipulation in BabyAI (Chevalier-Boisvert et al., 2019). They test information acquisition, prerequisite resolution, and spatial execution. We use 140 seen/134 unseen ALFWorld tasks, all 100 AgentGym TextCraft tasks (Xi et al., 2025), and 100 BabyAI instances balanced across five families, with action budgets 50/30/128. **Experience and evaluation.** Offline sources build artifacts rather than policy weights: ALFWorld uses training-split episodes, TextCraft excludes evaluation-goal overlap, and BabyAI separates source/test instances. Methods share evaluation tasks within each comparison; Appendix B gives source selection and protocols.

Policies and method labels. The primary comparison uses frozen Qwen3-4B-Instruct-2507. **P** (policy-only) uses current-task history; **PM** adds retrieved memory; **PMR** adds condition-checked recovery. Under a common protocol, we compare AWM (Wang et al., 2025), Synapse (Zheng et al., 2024), ReasoningBank (Ouyang et al., 2026), and AdaMEM-LOW/HIGH (Zhang et al., 2026). AdaMEM uses inference-time strategy control without Step-MFT, and BabyAI ReasoningBank uses static success-source memory. Appendix B.3 details implementations and source settings. **Metrics and study organization.** Success rate (SR) measures task completion; online actions/calls/tokens include failures and replaced proposals but exclude offline construction. We report means $\pm$ sample SD over three fixed-task runs (unseen Synapse: four), with cohort mappings in Appendix B.2. Local model-serving runs use a server equipped with five NVIDIA L40S GPUs; the serving configuration is held fixed within each comparison, and latency is reported separately as request wall-clock time.

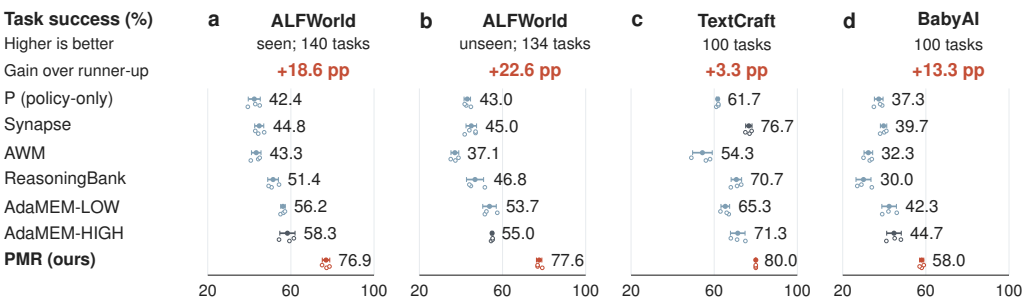

Figure 2: **Task success under the primary policy.** Means ± SD and individual runs ($n = 3$; b Synapse: 4). Red: PMR; dark gray: runner-up; references follow release order. All panels share a 20–100% axis. Baseline implementations and settings are detailed in Appendix B.3.

Table 1: **Online cost per task.** A: environment actions; C: model calls; K: input-plus-output tokens in thousands. Costs include failures and replaced proposals, exclude offline construction. Means over the runs in Figure 2; unseen PMR token costs use rounded totals. Per-run values: Appendix F.8.

	ALF. seen			ALF. unseen			TextCraft			BabyAI		
Method	A	C	K	A	C	K	A	C	K	A	C	K
P (policy-only)	37.2	37.2	109.6	37.0	37.0	118.2	16.5	16.5	**20.6**	90.7	90.7	64.5
Synapse	34.1	34.1	166.9	35.1	35.1	185.0	13.6	13.6	28.8	82.4	82.4	214.9
AWM	35.6	35.6	195.4	38.1	38.1	219.3	19.3	19.3	227.4	84.4	84.4	284.6
ReasoningBank	34.9	34.9	108.2	35.8	35.8	130.9	15.2	15.2	23.0	87.9	87.9	85.5
AdaMEM-LOW	31.1	67.3	202.7	31.9	69.5	229.0	15.2	33.0	63.2	76.4	157.8	228.7
AdaMEM-HIGH	31.7	58.1	243.3	32.2	63.2	272.0	14.4	18.4	43.0	75.7	127.4	231.4
PMR	**25.4**	**25.4**	**62.8**	**25.1**	**25.1**	**74.9**	**13.5**	**13.5**	23.0	**73.5**	**73.5**	**52.5**

4.2 Overall performance and online cost

Higher success across four task splits. PMR leads all four task splits (Figure 2) by 18.6/22.6 points on ALFWorld seen/unseen, 3.3 on TextCraft, and 13.3 on BabyAI. The runner-up is AdaMEM-HIGH except on TextCraft (Synapse), showing gains across the distinct recovery demands of all three environments. **Higher success need not require more online work.** PMR minimizes mean actions/calls on all splits and tokens on three (Table 1); TextCraft P uses fewer tokens at lower success. Thus the success gains do not require more online decision calls, consistent with the recovery channel's use of compiled support rather than auxiliary generation. Memory alone does not uniformly improve over P in these comparisons. **Cross-model evidence.** PMR leads TextCraft success in all six rounds under Llama-3.2-3B and Qwen2.5-3B, with the highest mean success (40.3/37.3%) and lowest tokens/task (22.5/21.7K; Figure 12).

Source-pool composition. To test whether reusable evidence depends on retaining unsuccessful source episodes, a PMR source-pool cohort compares a 300-episode mixed pool (150 successful and 150 unsuccessful) with a 300-success pool on the same 134 unseen tasks. The mixed pool reaches 74.1% mean success versus 66.9% for success-only construction (Table 8). The same three mixed-pool PMR runs are then paired with direct-effect-off evaluations in the fixed-bank intervention, allowing source composition and online execution to be read as separate questions rather than conflated (Table 4). The corresponding PMR records and AdaMEM-LOW/HIGH source-composition comparisons are reported together in Table 8, while the primary 77.6% PMR cohort remains the main comparison result.

4.3 Recovery-component configurations

Configurations and purpose. Table 2 compares **P** (policy only), **PM** (policy plus memory), and **PMR** (policy, memory, and recovery). **w/o direct-effect** removes direct goal-effect recovery; **w/o information** removes stagnation-triggered information recovery. Both retain retrieval, chains, and the

Table 2: **Recovery-component configurations.** 134 unseen ALFWorld tasks, budget 50, three-run mean ± SD. Actions equal model calls in this cohort; K denotes online tokens in thousands. Token means use rounded recorded totals. Complete records: Table 11.

Configuration	SR (%)	Actions/task	Tokens/task (K)
P	46.5 ± 2.3	35.1 ± 0.5	115.7 ± 1.3
PM	56.5 ± 3.0	33.0 ± 0.5	105.5 ± 1.1
w/o direct-effect	55.0 ± 2.6	32.4 ± 1.1	108.7 ± 3.4
w/o information	73.9 ± 1.3	26.0 ± 0.2	76.1 ± 0.7
PMR	$\mathbf{77.6 \pm 1.3}$	$\mathbf{25.1 \pm 0.3}$	$\mathbf{74.9 \pm 3.0}$

other branch. The 77.6% reference uses the same PMR runs as Figure 2; Appendix B.2 maps cohorts and Appendix D defines switches. **Recovery-enabled configurations outperform retrieval alone.** PM exceeds P by 10.0 points and PMR exceeds PM by 21.1 (Table 2); the direct-effect-disabled and information-disabled variants are 22.6/3.7 points below PMR. **The fixed-bank paired intervention isolates the online contribution of direct-effect execution.** It holds compiled evidence, retrieval index, tasks, and paired random seeds fixed while disabling only that branch. Across three paired mixed300 evaluations, PMR reaches 74.1% versus 60.7% without direct-effect execution, a 13.4-point task-success gain; the full controller also uses 19.3% fewer model calls and 32.2% fewer online tokens (Table 4). This is the direct end-to-end evidence that the corrective branch adds value beyond the evidence bank itself.

From components to mechanisms. Direct effects can precede a stall, whereas information recovery requires stagnation. We next connect performance to evidence through source provenance and controlled support removal at fixed checkpoints and deployed events.

4.4 From source evidence to recovery utility

Failed episodes supply distinct context evidence. Figure 3a identifies 22/69 context classes supported only by unsuccessful episodes. Thus success-only filtering loses distinct evidence of recovery needs, supporting outcome-decoupled construction (Section 3.2); the effects can have separate sources. **Relevant support changes local recovery.** At 22 fixed checkpoints, relevant removal eliminates matching support and reduces escapes from 22 to 5; equal-size unrelated removal preserves all 22 (Figure 3b). With executor/effects fixed and retrieval off, the 17 favorable and zero reverse pairs instantiate Equation 17: a 17/22 escape-rate gain with zero unchanged-coverage residual.

Support also matters in deployed-event continuations. In the mechanism-deployment cohort with 106/134 successful tasks, recovery activates on 64/134 tasks (direct-effect: 53). Across 17 information/chain events, relevant removal reduces local endpoints 15 → 3 and task successes

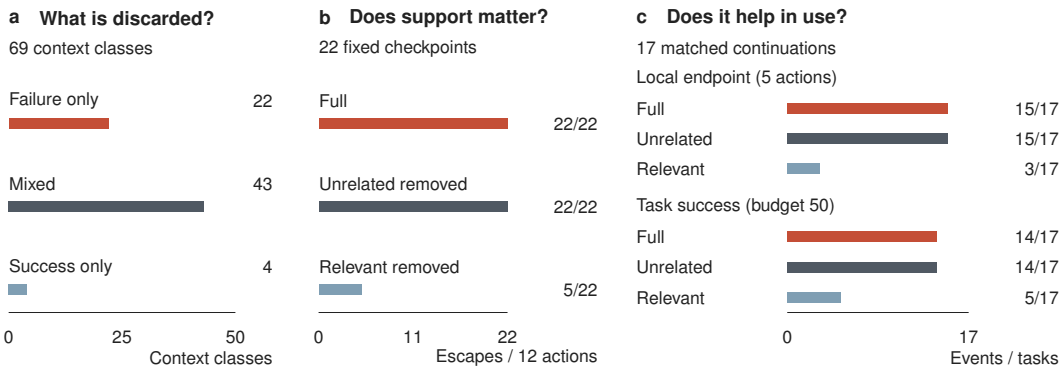

Figure 3: **From source evidence to recovery utility.** **(a)** Provenance of context evidence, distinct from effect evidence. **(b)** Confirmed pickup within 12 actions; executor/effects fixed, retrieval off. **(c)** Five-action local endpoints (information: sighting; chain: pickup) and task success within the 50-action total budget. Unrelated removal matches deletion size. Detailed analyses: Figures 5, 7, 8.

$14 \rightarrow 5$, with nine full-only wins and zero reverse wins; unrelated removal preserves outcomes (Figure 3c). Forks match history, state, and first proposal; their support spans 169 source tasks (Figure 7). In one knife-cleaning fork, full support reveals the knife after one action and finishes at action 19, whereas relevant removal fails at action 50; a mug-heating failure despite abundant support shows that matching alone is insufficient (Appendix E.5.3).

How reliably does a source pool retain support? Having tested support utility, we ask whether limited experience retains that support. Figure 4a shows median coverage rising from $8/22$ to $22/22$ checkpoints across 100–600 source episodes, with source-selection variation over 200 stratified nested draws. Figure 4b tests Proposition 1 below full-pool size: state/class retention predictions match observed frequencies with $2.22/2.38$-point RMSE. Source provenance therefore makes support availability predictable and inspectable under source retention in the fixed bank.

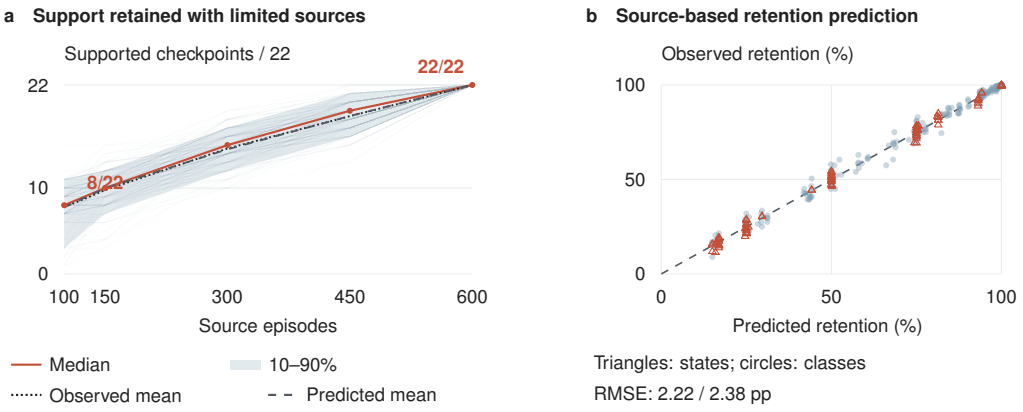

Figure 4: **Source coverage and retention prediction.** Fixed ALFWorld bank: 22 checkpoints, 69 classes. **(a)** Median and 10–90% spread over 200 stratified nested draws per size; dotted/dashed: observed/predicted means. **(b)** Predictions versus frequencies for 88 state–size and 276 class–size pairs below full-pool size. The spread measures source-selection sensitivity. Details: Figure 11.

What the evidence establishes. The experiments form a progressively stronger evidence chain. End-to-end comparisons establish gains in task success and online efficiency; component configurations separate retrieval from condition-checked recovery. Fixed-checkpoint and deployed-event interventions then hold the continuation protocol fixed while removing relevant support, showing that the observed gain follows the evidence required by the current obstacle rather than memory volume alone. Finally, source-retention analysis explains how source diversity determines whether that support remains available. Together, these studies connect PMR's representation, execution rule, and empirical behavior.

5 Conclusion

PMR shows that experience reuse need not be framed as retrieving a more similar task or replaying a complete trajectory. By separating local context evidence from effect evidence and separating policy advice from execution authority, PMR turns past interaction into conditional recovery opportunities that can be checked against the current state. Across multiple long-horizon interactive environments, PMR achieves higher task success with fewer test-time actions and model calls than strong baselines; the source-composition comparison and three-run fixed-bank intervention provide complementary evidence that the gain comes from retaining useful local evidence and executing it only when the current state supports it. More broadly, PMR suggests a design principle for fixed-policy agents: reusable experience should function as inspectable evidence for restoring progress, while the current environment remains the authority that decides whether and when a correction is valid.

AI USE STATEMENT

We used generative AI tools to aid and polish the writing of this paper. We reviewed the AI-assisted text and take responsibility for the final content, claims, and artifacts.

REFERENCES

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. BabyAI: A Platform to Study the Sample Efficiency of Grounded Language Learning. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://arxiv.org/abs/1810.08272.

Richard E. Fikes and Nils J. Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2(3–4):189–208, 1971. URL https://doi.org/10.1016/0004-3702(71)90010-5.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large Language Models Can Self-Correct with Tool-Interactive Critiquing. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=Sx038qxjek.

Qisheng Hu, Quanyu Long, and Wenya Wang. When Continual Learning Moves to Memory: A Study of Experience Reuse in LLM Agents. arXiv preprint arXiv:2604.27003, 2026. URL https://arxiv.org/abs/2604.27003.

Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander T Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Mengyuan Yan, Noah Brown, Michael Ahn, Omar Cortes, Nicolas Sievers, Clayton Tan, Sichun Xu, Diego Reyes, Jarek Rettinghouse, Jornell Quiambao, Peter Pastor, Linda Luu, Kuang-Huei Lee, Yuheng Kuang, Sally Jesmonth, Nikhil J. Joshi, Kyle Jeffrey, Rosario Jauregui Ruano, Jasmine Hsu, Keerthana Gopalakrishnan, Byron David, Andy Zeng, and Chuyuan Kelly Fu. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. In *Conference on Robot Learning (CoRL)*, pp. 287–318, 2023. URL https://proceedings.mlr.press/v205/ichter23a.html.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence*, 101(1–2):99–134, 1998. URL https://doi.org/10.1016/S0004-3702(98)00023-X.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. Dense Passage Retrieval for Open-Domain Question Answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, 2020. URL https://aclanthology.org/2020.emnlp-main.550/.

Jeonghye Kim, Sojeong Rhee, Minbeom Kim, Dohyung Kim, Sangmook Lee, Youngchul Sung, and Kyomin Jung. ReflAct: World-Grounded Decision Making in LLM Agents via Goal-State Reflection. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 33433–33465, 2025. URL https://aclanthology.org/2025.emnlp-main.1697/.

John E. Laird, Paul S. Rosenbloom, and Allen Newell. Chunking in Soar: The Anatomy of a General Learning Mechanism. *Machine Learning*, 1(1):11–46, 1986. URL https://doi.org/10.1007/BF00116249.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 9459–9474, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as Policies: Language Model Programs for Embodied Control. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9493–9500, 2023. URL https://doi.org/10.1109/ICRA48891.2023.10160591.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as Agents. In *International Conference on Learning Representations (ICLR)*, pp. 52989–53046, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/e9df36b21ff4ee211a8b71ee8b7e9f57-Abstract-Conference.html.

Yimeng Liu, Misha Sra, Jeevana Priya Inala, and Chenglong Wang. ReUseIt: Synthesizing Reusable AI Agent Workflows for Web Automation. In *ACM International Conference on Intelligent User Interfaces (IUI)*, pp. 885–908, 2026. URL https://doi.org/10.1145/3742413.3789083.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. AgentBoard: An Analytical Evaluation Board of Multi-turn LLM Agents. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 74325–74362, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/877b40688e330a0e2a3fc24084208dfa-Abstract-Datasets_and_Benchmarks_Track.html.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative Refinement with Self-Feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 46534–46594, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. ReasoningBank: Scaling Agent Self-Evolving with Reasoning Memory. In *International Conference on Learning Representations (ICLR)*, pp. 94327–94354, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/980ea04d23d1f6908964eba2a74afe45-Abstract-Conference.html.

Archiki Prasad, Alexander Koller, Mareike Hartmann, Peter Clark, Ashish Sabharwal, Mohit Bansal, and Tushar Khot. ADaPT: As-Needed Decomposition and Planning with Language Models. In *Findings of the Association for Computational Linguistics: NAACL*, pp. 4226–4252, 2024. URL https://aclanthology.org/2024.findings-naacl.264/.

Yuli Qiu, Yutong Li, Wei Su, Zeming Liu, Wanxiang Che, Heyan Huang, Haifeng Wang, and Yuang Guo. LifeMem: Enabling Lifelong Experience Reuse for LLM Agents. arXiv preprint arXiv:2609.12655, 2026. URL https://arxiv.org/abs/2609.12655.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. Learning To Retrieve Prompts for In-Context Learning. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pp. 2655–2671, 2022. URL https://aclanthology.org/2022.naacl-main.191/.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 8634–8652, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

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 (ICLR)*, 2021. URL https://arxiv.org/abs/2010.03768.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. ProgPrompt: Generating Situated Robot Task Plans Using Large Language Models. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11523–11530, 2023. URL https://doi.org/10.1109/ICRA48891.2023.10161317.

Theodore R. Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. Cognitive Architectures for Language Agents. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=1i6ZCvflQJ.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artificial Intelligence*, 112(1–2): 181–211, 1999. URL https://doi.org/10.1016/S0004-3702(99)00052-1.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An Open-Ended Embodied Agent with Large Language Models. *Transactions on Machine Learning Research*, 2024. URL https://rpl.cs.utexas.edu/publications/2024/03/14/wang-tmlr24-voyager/.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent Workflow Memory. In *International Conference on Machine Learning (ICML)*, pp. 63897–63911, 2025. URL https://proceedings.mlr.press/v267/wang25bx.html.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Xin Guo, Dingwen Yang, Chenyang Liao, Wei He, Songyang Gao, Lu Chen, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, Xipeng Qiu, Xuan-Jing Huang, Zuxuan Wu, and Yu-Gang Jiang. AgentGym: Evaluating and Training Large Language Model-based Agents across Diverse Environments. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 27914–27961, 2025. URL https://aclanthology.org/2025.acl-long.1355/.

Zidi Xiong, Yuping Lin, Wenya Xie, Pengfei He, Zirui Liu, Jiliang Tang, Himabindu Lakkaraju, and Zhen Xiang. How Memory Management Impacts LLM Agents: An Empirical Study of Experience-Following Behavior. arXiv preprint arXiv:2505.16067, 2025. URL https://arxiv.org/abs/2505.16067.

Qiang Yang, Kangheng Wu, and Yunfei Jiang. Learning Action Models from Plan Examples Using Weighted MAX-SAT. *Artificial Intelligence*, 171(2–3):107–143, 2007. URL https://doi.org/10.1016/j.artint.2006.11.005.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 11809–11822, 2023a. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/271db9922b8d1f4dd7aaef84ed5ac703-Abstract-Conference.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations (ICLR)*, 2023b. URL https://openreview.net/forum?id=WE_vluYUL-X.

Yunxiang Zhang, Yiheng Li, Ali Payani, and Lu Wang. AdaMEM: Test-Time Adaptive Memory for Language Agents. In *International Conference on Machine Learning (ICML)*, 2026. URL https://yunx-z.github.io/AdaMEM/.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM Agents Are Experiential Learners. In *AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024. URL https://ojs.aaai.org/index.php/AAAI/article/view/29936.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-Exemplar Prompting with Memory for Computer Control. In *International Conference on Learning Representations (ICLR)*, pp. 19036–19066, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/52f050499cf82fa8efb588e263f6f3a7-Abstract-Conference.html.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language Agent Tree Search Unifies Reasoning, Acting, and Planning in Language Models. In *International Conference on Machine Learning (ICML)*, pp. 62138–62160, 2024. URL https://proceedings.mlr.press/v235/zhou24r.html.

Deyu Zou, Yongqiang Chen, Jianxiang Wang, Garry Yang, Mufei Li, Qing Da, James Cheng, Pan Li, and Yu Gong. Reducing Belief Deviation in Reinforcement Learning for Active Reasoning of LLM Agents. In *International Conference on Learning Representations (ICLR)*, pp. 21511–21545, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/2468f84a13ff8bb6767a67518fb596eb-Abstract-Conference.html.

A COMPLETE CONSTRUCTION AND EXECUTION ALGORITHMS

The main text defines the evidence representation and execution rule. Algorithms 1 and 2 make their construction and state updates explicit as complete implementation specifications.

Algorithm 1 Offline conditional experience construction in ALFWorld

Input: Training episodes $\mathcal{D}$; interface Φ; fixed configuration λ
Output: Artifacts $(\mathcal{B}, \mathcal{M}, \mathcal{J})$ or BUILDFAILURE

$\mathcal{T} \leftarrow \mathrm{Project}_{\Phi}(\mathcal{D})$ ▷ *Outcome-free transition view*
$\mathcal{I} \leftarrow \mathrm{Discover}_{\Phi}(\mathcal{T}; \lambda_{\mathrm{disc}})$
$(\mathcal{E}, \mathcal{Q}) \leftarrow \mathrm{Induce}_{\Phi}(\mathcal{T}, \mathcal{I}; \lambda_{\mathrm{ind}})$
$\mathcal{P} \leftarrow \mathrm{Compose}(\mathcal{E}, \mathcal{Q}; d_{\max})$
if $\mathcal{P} = \varnothing$ **then**
  $p^{\star} \leftarrow \bot; \mathcal{C} \leftarrow \varnothing$ ▷ *Explicit empty-chain state*
else
  $p^{\star} \leftarrow \min_{\prec_{\lambda}} \mathcal{P}; \mathcal{C} \leftarrow \varnothing$ ▷ *One chain per build*
  for all $i \in \mathcal{I}$ **do**
    $p_i \leftarrow \mathrm{LongestCompatibleSuffix}_{\Phi}(i, p^{\star})$
    if $p_i \neq \bot$ **then**
      $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathrm{Link}_{\Phi}(i, p_i)\}$ ▷ *Keep both evidence roles*
    end if
  end for
end if
$\mathcal{B}_{\mathrm{aux}} \leftarrow \mathrm{AuxiliarySupport}_{\Phi}(\mathcal{T}, \mathcal{I}, p^{\star}; \lambda)$
if $\neg\, \mathrm{Pass}_{\Phi}(p^{\star}, \mathcal{C}, \mathcal{B}_{\mathrm{aux}}; \lambda)$ **then**
  return BUILDFAILURE ▷ *Required evidence checks failed*
end if
$\mathcal{B} \leftarrow \mathrm{Assemble}(p^{\star}, \mathcal{C}, \mathcal{B}_{\mathrm{aux}})$
$\mathcal{M} \leftarrow \mathrm{EventCards}_{\Phi}(\mathcal{D}, \mathcal{I}, \mathcal{B}; \lambda_{\mathrm{card}})$
$\mathcal{J} \leftarrow \mathrm{EmbedIndex}(\mathcal{M})$
return $(\mathcal{B}, \mathcal{M}, \mathcal{J})$

Algorithm 2 Stateful experience reuse under an action budget

Input: Goal g, initial observation o_0, frozen policy π_{θ}, interface Φ
Input: Frozen artifacts $(\mathcal{B}, \mathcal{M}, \mathcal{J})$, action budget B
Output: Interaction history h, execution trace $\mathcal{L}$

$h_0 \leftarrow (o_0)$; $(r_0, \eta_0) \leftarrow \mathrm{Initialize}_{\Phi}(g, o_0)$; $\mathcal{L} \leftarrow \varnothing$
for $t = 0, \ldots, B-1$ **do**
  $z_t \leftarrow \Phi(g, h_t)$; $m_t \leftarrow \varnothing$
  $(b_t, \bar{\eta}_t) \leftarrow \mathrm{RetrievalGate}_{\Phi}(z_t, r_t, \eta_t, t)$
  if $b_t = 1$ **then**
    $\mathcal{M}_t \leftarrow \{m \in \mathcal{M} : K_{\Phi}(m, z_t, r_t) = 1\}$
    $m_t \leftarrow \mathrm{Render}_{\Phi}(\mathrm{TopK}_{\mathrm{event}}(\mathcal{M}_t, \mathcal{J}, g, h_t))$
  end if
  $\widetilde{a}_t \sim \pi_{\theta}(\cdot \mid g, h_t, m_t)$ ▷ *Policy call is counted*
  $(u_t, \gamma_t, \bar{r}_t) \leftarrow \mathrm{Resolve}_{\Phi}(\mathcal{B}, z_t, r_t, \widetilde{a}_t)$
  $a_t \leftarrow \widetilde{a}_t$
  if $\gamma_t = 1$ **then**
    $a_t \leftarrow \mathrm{Action}(u_t)$ ▷ *Branch-specific checks passed*
  end if
  $(o_{t+1}, \mathrm{done}) \leftarrow \mathrm{EnvStep}(a_t)$ ▷ *Exactly one environment action*
  $h_{t+1} \leftarrow h_t \,\|\, (a_t, o_{t+1})$; $z_{t+1} \leftarrow \Phi(g, h_{t+1})$
  $r_{t+1} \leftarrow \mathrm{Observe}_{\Phi}(\bar{r}_t, z_t, a_t, z_{t+1})$; $\eta_{t+1} \leftarrow \bar{\eta}_t$
  $\mathcal{L} \leftarrow \mathcal{L} \,\|\, (t, z_t, u_t, \gamma_t, a_t, z_{t+1})$
  if done **then**
    return $(h_{t+1}, \mathcal{L})$
  end if
end for
return $(h_B, \mathcal{L})$

B ENVIRONMENTS, DATA SPLITS, AND EVALUATION PROTOCOLS

B.1 ENVIRONMENTS AND SOURCE–EVALUATION BOUNDARIES

The environments expose complementary prerequisites for progress: finding household objects, satisfying material dependencies, and executing spatial instructions. Together they test the same evidence-to-action principle under distinct state and action interfaces. Training denotes offline source experience and artifact validation; policy weights remain frozen.

ALFWorld. ALFWorld (Shridhar et al., 2021) provides textual observations and household actions. Evaluation uses the 140 seen and 134 unseen benchmark instances with a 50-action limit; source experience comes from the training split. The unseen set contains 24 place, 31 clean, 21 cool, 23 heat, 17 two-object, and 18 light-inspection tasks. The mechanism bank uses 600 source episodes, including 300 successful ones, with task-grouped source partitions for artifact validation (Appendix C).

TextCraft. TextCraft was introduced by ADaPT (Prasad et al., 2024); the agent obtains materials and crafts items using public inventory, quantities, and recipes. We use all 100 TextCraft evaluation instances distributed with AgentGym (Xi et al., 2025), with a shared 30-action limit. The source pool contains 192 successful and unsuccessful trajectories; its 153 successful trajectories supply the success-based reference memories. Sources exclude evaluation-goal and initial-instance overlap, and candidate discovery and validation are grouped within training tasks. This separation tests reuse of local dependencies rather than retrieval of an evaluated end-goal solution, while the complete evaluation list prevents outcome-based task selection.

BabyAI. BabyAI (Chevalier-Boisvert et al., 2019) tests instruction following through local observations and atomic spatial actions. Evaluation fixes 20 instances in each of five families: go-to, pickup, open, put-next, and pickup-then-go-to, with a 128-action limit. The five-family coverage and 128-step cap also appear in ReflAct (Kim et al., 2025), Appendix H; we double the instances per family and measure task completion. Test instances use indices 15–34 per family, fixed before observing outcomes and disjoint from source and prior reference instances. Equal family counts prevent family imbalance. The source pool contains 400 episodes, including 187 successes used by success-based memories.

B.2 POLICIES, BUDGETS, AND STUDY NAVIGATION

The primary policy is Qwen3-4B-Instruct-2507; TextCraft additionally evaluates Llama-3.2-3B and Qwen2.5-3B. Each comparison uses a common evaluation task set and environment action budget while retaining each method's prompts, source selection, and auxiliary-generation mechanism. TextCraft uses temperature 0.7, up to 4096 output tokens, a 32768-token context, and full public history. ALFWorld uses temperature 0.7, up to 2048 output tokens, a 32768-token context, and full history. BabyAI uses temperature 0.7, up to 2048 output tokens, and a 12-turn history. Method implementations are detailed in Appendix B.3. Local model-serving runs use the same five-GPU NVIDIA L40S server across each study; latency is reported as model-request wall-clock time under that serving configuration.

Study	Population and comparison	Results / definitions
Overall comparison	Four splits; seven methods; shared tasks and action limits within each split.	Section 4.2; Appendix F.8.
Recovery components	134 unseen ALFWorld tasks; P, PM, PMR, and two branch removals. Full SR: 77.6%.	Table 2; Table 11; Appendix D.
Fixed-bank execution	134 unseen tasks; 300 mixed source episodes; three paired runs, direct-effect on/off, identical compiled bank within each pair.	Table 4; Appendix E.6.
Source composition	134 unseen tasks; PMR source-pool and AdaMEM source-composition repeats.	Table 8; Appendix F.2.
Mechanism-analysis cohort	134 unseen ALFWorld tasks; mixed300 retrieval-only and PMR configurations, three repeats each; recovery-enabled reference SR: 74.1%.	Table 10.
Cross-model comparison	100 TextCraft tasks; two additional policies; three runs per method/policy.	Figure 12; Table 9.
Fixed-checkpoint intervention	22 ALFWorld checkpoints; 12-action horizon; relevant versus equal-size unrelated removal; executor/effects fixed, retrieval off.	Figure 5; Appendix E.5.1.
Deployed-event intervention	17 information/chain events from the 106/134 run; five-action local endpoint, 50-action total budget; three matched arms.	Figures 7–8; Appendix E.5.3.
Source retention	Fixed bank; 22 checkpoints and 69 classes; 200 nested draws at each of five source sizes, not policy reruns.	Figures 4, 11; Appendix E.5.6.
Search ordering	Separate 20-state held-out training cohort; evidence order versus five random orders per state.	Figure 6; Appendix E.5.2.

Reading across studies. Overall and component comparisons use 134 unseen ALFWorld tasks, a 50-action cap, and the same full-method PMR configuration. They share the three-run PMR reference with success counts $106/103/103$ ($77.6\% \pm 1.3$). Component controls use their designated run cohorts; the end-to-end branch ablations rebuild dependent artifacts, while the three-run fixed-bank intervention holds the compiled evidence and retrieval index fixed within each pair.

The mechanism-analysis cohort in Table 10 uses the $104/96/98$-success reference runs ($74.1\% \pm 3.1$) to anchor source-pool and mechanism analyses. The 77.6% cohort remains the main comparison result. Local support-removal studies use a fixed executor/effect catalog at selected checkpoints or matched event continuations from the $106/134$ mechanism-deployment cohort. Local endpoints, task success, and source retention are reported at their corresponding analysis units.

Metrics and repeats. SR is task success rate; steps/actions count submitted environment actions, including corrective and invalid actions; calls include auxiliary model requests; tokens count input plus output usage. Online means include failed episodes and exclude offline construction. Cap denotes action-limit attainment, not detected stagnation. Three repeated runs are reported per main comparison cell except unseen Synapse (four). Mean $\pm$ sample SD describes run variation, whereas retention bands describe source-subset variation. Exact records remain in Appendix F.

B.3 Baseline implementations and cost accounting

What is held common. Comparisons use the observations, evaluation task sets, and outer action limits in Appendix B.1; native termination may occur earlier. Policy weights remain frozen. Appendix B.2 specifies the policy and generation-setting scope. The comparison evaluates inference-time experience reuse under a common frozen policy and action budget. The table identifies the retained mechanisms and their scope.

Method	Experience supplied to the agent	Decision or execution route
P	Current task and interaction history; no cross-task memory	Policy-generated environment actions.
Synapse	Retrieved trajectory exemplars	Examples condition policy generation; no PMR recovery controller.
AWM	Offline-induced workflow memory	Workflows enter the action-generation context; workflow guidance is distinct from executing PMR's admitted effects.
ReasoningBank	Retrieved strategy entries; the BabyAI construction uses successful sources	Advice conditions the policy; the BabyAI adapter uses static retrieval rather than online memory evolution.
AdaMEM-LOW	Retrieved experience and a carried short-term strategy	Initial strategy, cross-step retention, and requested refresh; inference-time adaptation without Step-MFT.
AdaMEM-HIGH	Retrieved experience and request-triggered strategy synthesis	A memory request triggers strategy synthesis and another action decision; no Step-MFT.
PMR	Event-bounded cards plus separately compiled structural support	Cards advise the policy; recovery replaces a proposal only after current-state admission checks.

Environment-specific construction. Each method uses the public environment interface described in Appendix B.1, with source and evaluation tasks separated. The TextCraft source pool contains 192 trajectories, including 153 successes available to success-based memories. The BabyAI source pool contains 400 episodes, including 187 successful episodes used by success-based memories. Synapse and ReasoningBank retrieve top-1 from the task and initial observation and reuse the selected experience within the episode; the BabyAI ReasoningBank adapter uses its static successful-source retrieval path.

Adaptive strategies and generation constraints. LOW names the strategy-refresh logic rather than the realized call count. We retain inference-time retrieval and strategy control without Step-MFT to keep policy weights fixed. In BabyAI, AdaMEM uses bounded structured responses and ReasoningBank uses repetition penalty 1.05; associated settings are temperature 0.7, a 12-turn history, and a 2048-token output cap. Appendix B.2 records the effective settings for each study cohort.

Online cost accounting. For N tasks, K successes, and recorded totals S actions, C model calls, and T tokens, per-task costs are $S/N, C/N, T/N$. Amortized cost per solved task divides the same totals by K, including failures. Calls include action proposals, strategy generation, and replaced proposals; tokens include their recorded input/output usage. The tables report online policy and strategy costs, with offline construction and embedding outside this accounting. Predicate definitions, grounding templates, terminal tests, and branch arbitration are implementation inputs detailed in Appendix C.

Interpretation of the comparison. All methods share evaluation tasks and action limits while retaining their respective memory-construction and deployment mechanisms. This design compares complete experience-reuse systems under a common protocol; the component, fixed-executor, and fixed-bank interventions separately analyze recovery architecture, support relevance, and online execution.

C Implementation and execution details

C.1 Evidence extraction and admission criteria

The environment interface defines predicates, goal obligations, progress events, action grounding, and controller ordering. Within this shared representation, training trajectories determine the retained effects, recovery prefixes, source associations, and support statistics; fixed admission criteria and execution limits govern their use.

ALFWorld groups transitions into intervals without goal-relevant progress. The default detector requires at least two occurrences for recurrent states, repeated no-event actions, or repeated inad-

missible actions. Records containing only invalid-action stalls are retained separately and do not have the same eligibility as recurrence- or repetition-supported relational records. For the latter, the first observable recurrence or repetition determines the trigger. Future recovery observations are available during offline induction but not when matching a condition online.

Training partitions are determined by task signature, keeping repeated episodes of a task together. The ALFWorld single-transition candidate filter requires at least five positive observations in the fit partition, one in the held-out training partition, and empirical effect consistency of at least 0.90 in each. Support is counted in transition observations, with repeated episodes grouped by task. Bounded-prefix candidates require at least five legal positive prefixes in the fit partition and one in the held-out training partition. They record opportunity counts but do not apply the single-transition 0.90 rate threshold. Direct effects and device bindings follow separate admission procedures.

For single-transition induction, let $N_d(\kappa, a)$ count eligible attempts in context κ with action schema a in training partition d, and let $N_d(\kappa, a, e)$ count those exhibiting effect e. The empirical consistency rate is

$$\widehat{p}_d(e \mid \kappa, a) = \frac{N_d(\kappa, a, e)}{N_d(\kappa, a)}, \qquad N_d(\kappa, a) > 0. \tag{5}$$

Eligibility requires an outstanding obligation, a legal candidate action, and a context represented in impasse discovery. An eligible attempt without e contributes to the denominator and can prevent admission; invalid actions are excluded. The threshold is an empirical admission criterion estimated from task-grouped source transitions, so episode-level failure does not erase an observed local effect. Deployment combines this evidence with current-state grounding and observation-confirmed terminals.

C.2 Composition, records, and contextual cards

ALFWorld constructs candidate paths by unifying observed relation additions with effect prerequisites. The default maximum composition depth is three. Its path order first prefers shorter paths, then an early observable milestone, a shorter median observed terminal length, higher held-out and fit support, and a deterministic identifier tie-break. Algorithm 1 selects one path per ALFWorld build, and compatible contexts link to suffixes of that shared sequence. A context record stores provenance and effect identifiers; effect entries store prerequisites, terminals, budgets, and induction statistics. Contextual cards preserve event boundaries and source links and require a successful source episode associated with a recovery contract, while the structured discovery pool uses both outcomes. Rebuilding from a restricted source pool recomputes these dependencies; direct effects remain separately admitted.

Context diversity and executable recoveries. The ALFWorld deployment analyzed in Figure 7 retains one two-stage chain and its one-stage suffix. Its 613 context-source records comprise 63 full-chain and 550 suffix records, spanning 451 source episodes and 413 task signatures. The local mechanism bank used for the 22-checkpoint and source-retention analyses has 684 records ($63/621$ by the same sequence split), spanning 504 episodes and 460 task signatures. These separate banks quantify context-source diversity over two executable sequences. The 69 coarse source-analysis classes summarize context diversity; grounding maps their linked sequences to current objects, while direct-effect and information recovery provide separate routes.

Cards are first filtered for compatibility and then ranked by cosine similarity between the query embedding and card embeddings. Selection deduplicates event signatures. Initial retrieval is gated by outstanding obligations; later retrieval additionally uses the available verified recovery context and per-episode/per-state limits. The exact schedule and limits are configuration parameters. The raw-fragment control renders the same selected event segment without its card presentation: it preserves retrieval identity and event boundaries and is not an ablation of event segmentation.

C.3 Execution invariants and resource accounting

Feedback and control representations. Model-based refinement (Madaan et al., 2023) and tool-interactive critique (Gou et al., 2024) distinguish internal feedback from external checking. PMR defines controller completion through observation-derived predicates. Programmatic control (Liang

et al., 2023) and assertion-checked plans (Singh et al., 2023) provide related execution abstractions; PMR compiles its control structure offline from experience-supported candidates and applies fixed adapter checks online. This execution path complements reasoning-branch expansion in Tree of Thoughts (Yao et al., 2023a) and online action-tree search in LATS (Zhou et al., 2024).

Invariant 1 (Observation-grounded stage completion)**.** Fix one activation, with grounded stage terminals $T_1, \ldots, T_L$. Assume that the only operation recording stage j as complete tests T_j on an actual observation-derived state under the bindings of that stage; no prediction, timeout, or release is marked as completion. Let J_k denote stages recorded complete after controller update k, and let O_k be the observation indices available by that update. With bindings recorded at each completion, the invariant is

$$j \in J_k \quad \Longrightarrow \quad \exists t \in O_k : \; T_j(z_t; \text{bindings at completion}) = 1. \tag{6}$$

Proof. Initially J_0 is empty. An update that only records feedback, reserves budget, releases control, or retains the current stage adds no completed stage. An update that records completion has just tested the grounded terminal on available observation z_t, providing the witness for every new member of J_{k+1}. Previously completed stages retain their historical witnesses. Induction establishes the invariant. The check can occur in the next pre-action resolution after feedback is ingested; it need not be performed inside the observation-update operator itself. A stage whose terminal already holds is recorded as satisfied under the current observation. □

This invariant guarantees that every recorded stage completion has an observation witness under its grounded bindings. Local escape and terminal task success are evaluated separately with the environment-level criteria in Appendix E.

Per-activation action accounting. Fix one chain activation. Assume that every corrective action is charged to exactly one stage, each stage j starts with integer budget $b_j > 0$, charging reduces its remaining budget by one, no action can be issued when that budget is zero, and neither budgets nor stage indices reset within the activation. Let $n_j(k)$ count actions charged to stage j after k corrective actions. Initially $n_j(0) = 0$. If the next action is charged to stage j, its admission requires $n_j(k) < b_j$, so $n_j(k+1) = n_j(k) + 1 \le b_j$; all other counters remain unchanged. Induction yields $0 \le n_j(k) \le b_j$ for every stage and every prefix. Since each action is counted once,

$$N_{\text{corr}} = \sum_{j=1}^{L} n_j \le \sum_{j=1}^{L} b_j. \tag{7}$$

A completed stage is not revisited; early release or episode termination can only decrease the count. If the activation begins with B_{rem} episode actions remaining, the shared budget further gives $N_{\text{corr}} \le \min\{B_{\text{rem}}, \sum_j b_j\}$. A later activation begins a new accounting interval. The bound controls corrective-action exposure; empirical sections report whether bounded interventions restore progress. □

Generative-request accounting. Consider a normally completed run of the stated algorithm, with one generative request per sampling operation, no request retries, and no failed iteration that generates a proposal but does not submit an action. Initialize policy-call and submitted-action counters to zero. Each executed iteration contains exactly one policy sampling operation and exactly one environment step, irrespective of the admission bit γ_t. Replacement changes the submitted action, not the number of policy requests. By induction the two counters coincide after every completed iteration. The loop either terminates after an environment step or reaches B iterations; hence both equal $T \le B$.

The retrieval, resolver, and observation-update operators are defined without auxiliary generative requests. Their auxiliary-generation count is initially zero and unchanged by each iteration, giving

$$N_{\text{policy}} = T, \qquad N_{\text{aux,gen}} = 0. \tag{8}$$

With one completed policy request per action, policy calls equal submitted actions and auxiliary generation is zero. If request t consumes ℓ_t^{in} input and ℓ_t^{out} output tokens, total policy tokens are $\sum_{t=0}^{T-1}(\ell_t^{\text{in}} + \ell_t^{\text{out}})$. Section 4.2 and Appendix F report realized calls, tokens, and latency under the measured serving configurations. □

C.4 Online branches and arbitration

ALFWorld checks effect-chain recovery and direct-obligation execution before initiating a new information-recovery intervention. An active information controller retains its execution phase. Chain candidates match their effect sequence and current projected relation context. Direct execution uses independently admitted goal effects and its declared target-selection and action-grounding order. Information recovery additionally uses a repeated action with no observed event. Thus, an impasse-centered discovery process supports proactive as well as reactive execution. Chain contracts use exact equality over canonical projected relation states and requested effect identifiers; direct effects use their own support and applicability checks. The gate in Equation 3 abbreviates these branch-specific checks and includes controller state explicitly.

An active chain rechecks prerequisites under its grounded bindings before issuing another action. Observed terminals advance its stage; each stage has a positive action bound. Direct execution separately tracks its action and stagnation limits. Failed-state retry suppression records the relevant state projection and blocks repeated chain activation while the same key persists; policy fallback remains available. The policy continues to supply proposals through the implemented calling protocol, and model calls made during recovery remain part of cost accounting. Offline support and online retry state are tracked separately in the mechanism studies.

C.5 Environment-specific discovery

TextCraft extends discovery with observed prefixes toward a missing output of an attempted action. Public inventory, quantities, recipe descriptions, and acquisition evidence support candidate grounding; failed attempts do not establish that the intended output was produced. BabyAI constructs local action-sequence candidates and checks their effects on separate training instances through its observation interface. The three environments share the same evidence-organization and admission discipline through environment-native candidate generators. Exact budgets, split procedures, and adapter predicates appear in their protocols.

C.6 Relationship to prior approaches

Learning from failed experience and reusing guarded behaviors have established precedents. ReasoningBank distills reasoning strategies from self-judged successful and failed experiences (Ouyang et al., 2026); AdaMEM synthesizes and refreshes strategy memory from trajectory experience (Zhang et al., 2026); AWM induces reusable workflows to guide generation (Wang et al., 2025). ReUseIt derives workflow checks and recovery actions from varied successful and failed runs, then evaluates checks through model-based inspection during web execution (Liu et al., 2026). PMR integrates these concerns at a new interface: observable impasse contexts are linked to separately supported local effects, compiled into context-conditioned candidates, and authorized against the current state alongside event-bounded retrieval. Its predicates, admission criteria, provenance, and execution checks make specific source support inspectable. T3 studies belief deviation and trajectory truncation for reinforcement-learning credit assignment (Zou et al., 2026), whereas PMR targets deployment-time recovery with fixed policy parameters. Soar's impasse-driven procedural learning is a classical antecedent (Laird et al., 1986). Memory addition/deletion can expose harmful experience-following (Xiong et al., 2025); representation and retrieval choices shape transfer and forgetting (Hu et al., 2026); and LifeMem distills textual skills from workflow clusters (Qiu et al., 2026). Overall comparisons evaluate complete baseline implementations under the common protocol, while fixed-checkpoint and fixed-bank interventions isolate support relevance and online recovery execution.

D Ablation terminology and interventions

D.1 Reported component configurations

P denotes policy-only execution with current-task history, not a trainable-versus-frozen distinction: every policy in this comparison is frozen. PM adds memory retrieval; PMR adds the admitted recovery branches available in the corresponding adapter. Table 2 uses the following configurations.

Paper label	Operational meaning
P (policy-only)	No cross-task experience retrieval; no recovery execution.
PM (retrieval only)	Retrieved contextual advice is supplied to P; recovery execution is disabled.
PMR (full)	Retrieval plus condition-checked recovery; ALFWorld includes direct-effect, chain, and information branches.
w/o direct-effect	Disable direct goal-effect intervention; retain retrieval, chain, and information recovery.
w/o information	Disable information recovery; retain retrieval, direct effects, and chains.

These end-to-end configurations measure recovery-enabled designs with their corresponding artifact graphs. The three-run paired fixed-bank intervention additionally holds compiled evidence and retrieval fixed while disabling only online direct-effect execution. Cohorts and result locations are mapped in Appendix B.2.

D.2 Additional probes and implementation controls

The following vocabulary describes supplementary interventions, not extra rows of task-level results in Table 2. The reported raw-fragment, goal-filter, and repetition probes are discussed in Appendix F.1.2; other controls specify local or implementation checks.

Control	Change from the corresponding full configuration
w/o experience retrieval	Retain recovery execution; disable retrieved policy context.
Raw event fragments	Render the same selected source segments without card presentation; preserve retrieval identity and event boundaries.
w/o goal compatibility filter	Remove goal/type/event filtering; retain ranking, deduplication, and retrieval schedule.
w/o recovery-context matching	Relax relation-context matching; retain other applicability, legality, and budget checks.
w/o multi-step recovery	Disable composed effect chains; retain direct-effect and information branches.
Action-repetition discovery	Replace impasse selection with consecutive identical executed actions and rebuild dependent artifacts.

Repetition changes discovery and downstream artifacts, while independent artifacts retain their original construction. Budget, retry-suppression, prerequisite, terminal, and handback probes test implementation behavior; same-state support removal tests local evidence utility; and the three-run fixed-bank intervention isolates online direct-effect execution. Together, these interventions provide complementary views of PMR's architecture.

E Mechanism analysis and empirical estimands

E.1 Notation and operator semantics

The notation below connects Algorithms 1–2 to the computational stages used in support-removal and local-recovery analyses.

Symbol	Meaning
$\mathcal{D}, \tau_i, \sigma_i, y_i$	Training pool; episode, task identity, final outcome.
Φ, λ	Environment interface and fixed algorithm configuration.
$\mathcal{T}, \mathcal{I}$	Outcome-free transition view and discovered impasse records.
$\mathcal{E}, \mathcal{Q}$	Admitted transition effects and bounded progress options.
$\mathcal{P}, p^\star, \prec_\lambda$	Candidate paths, selected path, deterministic selection order.
$\mathcal{C}, \mathcal{B}_{\mathrm{aux}}$	Conditional experience records (contracts) and auxiliary branch support.
$\mathcal{B}, \mathcal{M}, \mathcal{J}$	Structured artifacts, textual cards, dense retrieval index.
$z_t = (F_t, U_t, \mathcal{A}_t)$	Observable facts, outstanding obligations, action interface.
$r_t, \bar{r}_t$	Controller state before and after pre-action resolution.
$\eta_t, \bar{\eta}_t$	Retrieval scheduling state before and after its gate.
$m_t, \widetilde{a}_t$	Temporary retrieved text and frozen-policy proposal.
u_t, γ_t, a_t	Recovery proposal, admission bit, submitted action.
$d_{\max}, B, H_{\mathrm{loc}}$	Composition depth, episode action limit, local analysis horizon.

In Algorithm 1, Induce_Φ applies the distinct transition and prefix criteria in Appendix C; $\mathrm{Compose}$ enumerates the declared effect paths. $\mathrm{AuxiliarySupport}_\Phi$ covers direct-effect induction and information-recovery support, including their required training-side validation artifacts. Pass_Φ rejects failed required gates even when the chain is empty. $\mathrm{EventCards}_\Phi$ retains the successful-source and event-link requirements of the implementation. In Algorithm 2, $\mathrm{RetrievalGate}_\Phi$ updates seen/retrieved-state records and request counts, including steps on which no retrieval occurs. $\mathrm{Resolve}_\Phi$ performs the implemented branch arbitration and checks atomically. It returns the resulting pre-action state $\bar{r}_t$, including releases, retry blocking, or budget reservation as applicable. $\mathrm{Observe}_\Phi$ ingests actual feedback; a subsequent resolution may advance a stage using that evidence. Consequently an action is not charged twice, and no stage advances on an unobserved predicted effect. The policy proposal is generated before resolution even during an active recovery.

E.2 Relevant support, source classes, and authorization

Let x be a fixed decision checkpoint containing the restorable environment state, goal and history, controller state, retrieval state, and current policy proposal. Paired replay restores this checkpoint, while the policy and matcher receive only $z = \Phi(g, h)$. For a fixed branch query, define its matching support

$$\mathcal{R}_{\mathcal{C}}(x) = \{c \in \mathcal{C} : \mathrm{Match}_\Phi(c, x) = 1\}, \qquad S_{\mathcal{C}}(x) = \mathbf{1}[\mathcal{R}_{\mathcal{C}}(x) \neq \varnothing]. \tag{9}$$

The match includes the queried effect identifiers, canonical relation context, and any branch-specific obligation constraint. $S_{\mathcal{C}}$ is structural coverage, distinct from the admission bit $\gamma(x)$: matching support alone need not satisfy grounding, preservation, or budget checks. Cross-task support is the subset with source-task identity different from that of x.

Let $q_\Phi(c)$ be the complete matching key for this fixed branch, and $[c]_q = \{c' : q_\Phi(c') = q_\Phi(c)\}$ its equivalence class. A targeted deletion removes every member of the relevant classes, not just one record identifier. For a class q, write n_q^+ and n_q^- for the numbers of context-source episodes with final outcome success and failure. The indicator

$$F(q) = \mathbf{1}[n_q^+ = 0 \ \wedge \ n_q^- > 0] \tag{10}$$

identifies failure-only context support in the frozen pool. $F(q)$ records context-source provenance; effect provenance remains represented separately by $\mathcal{V}_c$. Source redundancy is $D(q) = |\{\sigma(c) : q_\Phi(c) = q\}|$, counting distinct source tasks rather than duplicated contracts.

E.3 Escape time and paired effects

Fix a local escape criterion ψ_x before observing branch outcomes, with x initially satisfying the blocked condition. It tests environment evidence separately from the controller's stage-completion flag. Let v denote a controlled memory variant, such as full support, related-support deletion, or

unrelated-support deletion. Index subsequent steps from the checkpoint. With randomness ω, define

$$\tau_v(x,\omega) = \inf\{k \geq 1 : \psi_x(o^v_{0:k}, a^v_{0:k-1}) = 1\}, \tag{11}$$

$$Y_{v,H_{\mathrm{loc}}}(x,\omega) = \mathbf{1}[\tau_v(x,\omega) \leq H_{\mathrm{loc}}], \qquad T_{v,H_{\mathrm{loc}}}(x,\omega) = \min\{\tau_v(x,\omega), H_{\mathrm{loc}}\}. \tag{12}$$

Set $\tau_v = \infty$ when escape never occurs; lack of escape by H_{loc} is right-censored at H_{loc}. A terminal failure does not remove the case from analysis: if episode termination occurs first, its outcome is carried forward for this measurement. Escape exactly at step H_{loc} is distinguished by $Y_{v,H_{\mathrm{loc}}}$ even though its capped time equals that of a non-escape.

For a fixed case distribution μ and stated randomness coupling, the survival curve is $S_v(k) = \Pr_{x\sim\mu,\omega}(\tau_v > k)$. The finite-horizon identity

$$\mathbb{E}[T_{v,H_{\mathrm{loc}}}] = \sum_{k=0}^{H_{\mathrm{loc}}-1} S_v(k) \tag{13}$$

follows by writing each integer-valued capped time as $\sum_{k=0}^{H_{\mathrm{loc}}-1} \mathbf{1}[\tau_v > k]$. Thus, area between survival curves equals a difference in restricted mean action counts. Paired contrasts are $\Delta_Y = \mathbb{E}[Y_{v,H_{\mathrm{loc}}} - Y_{w,H_{\mathrm{loc}}}]$ and $\Delta_T = \mathbb{E}[T_{w,H_{\mathrm{loc}}} - T_{v,H_{\mathrm{loc}}}]$, with positive values favoring v. In the ordering study, v is the evidence-based order and w a random order; in the support study, v is full support and w relevant-support removal with $H_{\mathrm{loc}} = 12$. Each controlled study fixes its checkpoint cohort, horizon, and continuation protocol.

E.4 Coverage prediction and utility analysis

Shared objects and experimental units. A checkpoint x is the complete replay checkpoint defined above. For a retained bank $\mathcal{C}_v$, write $s_v(x) = S_{\mathcal{C}_v}(x)$ using Equation 9; structural coverage is distinct from live admission and escape. In Proposition 1, $\mathbf{n}$ indexes retained source-episode counts. Source episodes are the sampling units, source tasks describe provenance diversity, and checkpoints are the continuation-evaluation units.

Proof of Proposition 1. Fix the compiled bank and its assignment of each record to a source episode, as in the source-retention experiment. For checkpoint x, let $D_h(x)$ be the set of supporting source episodes in stratum h, of size $d_h(x)$. Multiple matching records in one episode contribute one element, not multiple independent witnesses. A uniform size-n_h subset has $\binom{N_h}{n_h}$ possible selections. Precisely $\binom{N_h - d_h(x)}{n_h}$ of these avoid every supporting episode. Thus

$$\Pr\{D_h(x) \text{ is missed} \mid n_h\} = \frac{\binom{N_h - d_h(x)}{n_h}}{\binom{N_h}{n_h}}. \tag{14}$$

Conditional independence of the within-stratum selections gives the product probability of missing support in every stratum. Its complement proves Equation 4. For fixed checkpoints $x_1, \ldots, x_M$, linearity of expectation gives

$$\mathbb{E}\left[\sum_{i=1}^{M} s_{\mathbf{n}}(x_i)\right] = \sum_{i=1}^{M}\left[1 - \prod_h \frac{\binom{N_h - d_h(x_i)}{n_h}}{\binom{N_h}{n_h}}\right]. \tag{15}$$

Checkpoints can share support and hence have dependent coverage indicators; no independence assumption across them is used. If quotas are random, condition on them first and then average the conditional prediction. □

What the retention law says about unsuccessful sources. If all witnesses for x lie in unsuccessful-source strata and their quotas are zero, every factor in the no-support probability equals one; hence coverage is zero. Retaining more episodes in any stratum cannot lower coverage under a nested-selection coupling. For one stratum with d witnesses, the marginal increment, for $0 \leq n < N$, is

$$p(n+1) - p(n) = \frac{\binom{N-d}{n}}{\binom{N}{n}} \frac{d}{N-n}, \qquad p(n) = 1 - \frac{\binom{N-d}{n}}{\binom{N}{n}}. \tag{16}$$

The increment is nonnegative: a new draw supplies coverage exactly when earlier draws missed every witness. The law is exact for episode retention in a fixed compiled bank under stratified uniform sampling. The nested source draws approximate this sampling design; agreement with the formula tests predicted structural coverage from source redundancy.

Coverage–utility accounting. For two fixed banks m, n with $s_m(x) \geq s_n(x)$ and paired local-escape outcomes $Y_m, Y_n \in \{0, 1\}$, define the newly covered set $\mathcal{N}_{nm} = \{s_n = 0, s_m = 1\}$ and the unchanged-coverage residual $\mathcal{R}_{nm} = \mathbb{E}[\mathbf{1}_{\mathcal{N}_{nm}^c}(Y_m - Y_n)]$. Partitioning checkpoints by $\mathcal{N}_{nm}$ gives the exact identity

$$\mathbb{E}[Y_m - Y_n] = \Pr(\mathcal{N}_{nm}, Y_m = 1, Y_n = 0) - \Pr(\mathcal{N}_{nm}, Y_m = 0, Y_n = 1) + \mathcal{R}_{nm}. \tag{17}$$

For the fixed intervention below, m denotes the full-support bank and n the relevant-removal bank. Thus $\mathcal{N}_{nm}$ contains checkpoints whose support is newly available in the full arm, while $\mathcal{R}_{nm}$ captures changes where coverage is unchanged. Indeed,

$$\mathbb{E}[Y_m - Y_n] = \mathbb{E}[\mathbf{1}_{\mathcal{N}_{nm}}(Y_m - Y_n)] + \mathbb{E}[\mathbf{1}_{\mathcal{N}_{nm}^c}(Y_m - Y_n)] \tag{18}$$

$$= \Pr(\mathcal{N}_{nm}, Y_m = 1, Y_n = 0) - \Pr(\mathcal{N}_{nm}, Y_m = 0, Y_n = 1) + \mathcal{R}_{nm}. \tag{19}$$

In the fixed 22-checkpoint intervention, the full arm has matching cross-task support at every checkpoint. The relevant-removal arm first excludes target-task contracts and then deletes every cross-task contract returned by the matcher, so coverage switches from one to zero for all 22 checkpoints. Hence $\mathcal{N}_{nm}$ is the complete cohort and $\mathcal{R}_{nm} = 0$ by construction. The observed 17 favorable and zero reverse discordances instantiate Equation 17 as an escape-rate gain of $17/22$. In the separate 17-event continuation study, favorable/reverse discordances are $12/0$ for local endpoints and $9/0$ for terminal task outcomes under its matched protocol.

The finite-horizon identity in Equation 13 supplies a separate time estimand. It follows pointwise from $\min(\tau, H_{\mathrm{loc}}) = \sum_{k=0}^{H_{\mathrm{loc}}-1} \mathbf{1}[\tau > k]$, including $\tau = \infty$, followed by expectation over the finite sum. For arms v, w under the same checkpoint distribution and horizon, this also gives

$$\mathbb{E}[T_{w,H_{\mathrm{loc}}} - T_{v,H_{\mathrm{loc}}}] = \sum_{k=0}^{H_{\mathrm{loc}}-1} \big(S_w(k) - S_v(k)\big). \tag{20}$$

This measures actions saved within the local horizon, including capped non-escapes. Two arms can have the same escape rate by the horizon yet differ in when their escapes occur; conversely, escape at the final step and non-escape have the same capped cost. Reporting both the binary endpoint and capped time therefore distinguishes local effectiveness from speed. The separate ordering study uses these quantities to assess search efficiency, whereas deployed-event continuations additionally measure eventual task success under the remaining episode budget.

E.5 Mechanism experiments: support, execution, and transfer

The experiments below connect the evidence roles in Figure 1 to measured recovery behavior. Figures 5–8 test local utility, search ordering, and deployed-event continuations; Figures 9–11 examine source breadth, transfer, and retention. The analyses use benchmark tasks for end-to-end performance, fixed checkpoints for support interventions, and deployed events for matched continuations; each result is reported at its corresponding unit, with complete cohorts and source provenance retained in the figures.

E.5.1 Relevant support at fixed checkpoints

We use 22 ALFWorld checkpoints selected outcome-blind from the training pool: the first state with available cross-task support per selected task, excluding that task's own source contracts. Selection yields 16 states shared with action-repetition discovery and six additional typed-context states. Each checkpoint is evaluated with the same executor and admitted effect catalog, while retrieval is disabled. Escape means a confirmed pickup within 12 actions; full support is compared with removal of all matching support-class members and equal-size unrelated removal.

Relevant support changes recovery under controlled removal. The decisive measurement is the paired removal (Figure 5a). With full support, $22/22$ states escape within 12 actions. Deleting every

member of the matching equivalence class removes matching support at all 22 checkpoints and reduces escape to $5/22$. Equal-size removal of unrelated classes preserves matching support and all 22 escapes. Thus the paired coverage switch yields 17 favorable and zero reverse escape discordances, exactly instantiating Equation 17; the favorable pairs span seven of ten condition groups. On the six typed-only states, relaxed chain-relation matching with the same admitted executor also escapes all six, showing that typed matching expands discovery coverage while the execution checks retain local utility.

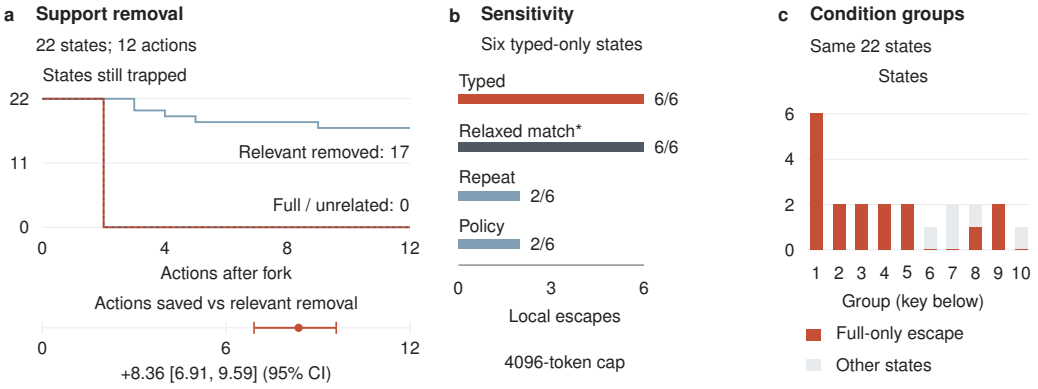

Figure 5: **Local utility and sensitivity of recovery support.** **(a)** States still trapped in the fixed 22-state catalog; escape is a confirmed pickup within 12 actions. Executor/effect catalog are fixed; retrieval is disabled. Full and unrelated-removal curves coincide. Full support saves 8.36 capped actions versus relevant removal (paired-case bootstrap 95% CI $[6.91, 9.59]$; non-escape costs 12). **(b)** Six typed-only states, 4096-token cap; *relaxed chain-relation matching with safety checks retained, extending discovery coverage to all six. **(c)** Condition groups: 17 full-only escape pairs, five other states, zero reverse pairs.

Condition groups. Groups 1–10 in panel c index coarse context groups; they summarize related checkpoints rather than independent experiments.

E.5.2 Coverage and evidence-based search ordering

Coverage grows with the pool, and so does escape. Filtering the compiled library to nested source subsets of 150, 300, and 600 episodes while holding the executor fixed, the number of blocked states with matching support rises $9 \to 12 \to 22$ out of 22, and the number that escape within the horizon rises $13 \to 14 \to 22$ (Figure 6a). Escape can exceed support because the frozen policy can also recover; the two counts converge at the full pool. Coverage sensitivity to *which* episodes are drawn is measured over 200 outcome- and family-stratified draws per tier: median supported states rise $8, 10, 15, 19, 22$, and empirical means track the conditional finite-bank prediction of Proposition 1 to within 0.25 at every tier (Figure 4; state/class detail in Figure 11).

Evidence-based ordering improves search efficiency. The evidence-based path order specified in Appendix C is tested separately from support removal. On a held-out cohort of 20 blocked training states, ordering candidate continuations by compiled evidence saves a mean of 2.43 capped actions per state against five random orders per state, with a task-paired bootstrap 95% interval of 0.82–4.07 (Figure 6b–c). Evidence ordering reaches eventual target sighting on 75% of states versus 59% under random ordering; its principal measured advantage is the reliable action-efficiency gain. Equation 13 relates this capped-time contrast to the area between escape survival curves.

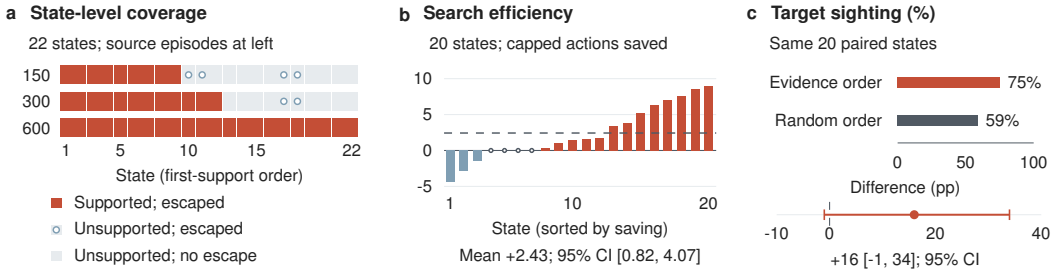

Figure 6: **State-level coverage and search efficiency. (a)** All 22 states across 150/300/600 source episodes, ordered by first support; fixed executor/effect catalog. Source-support subsets of one compiled bank; unsupported states can escape through the policy. **(b–c)** Separate 20-state held-out training cohort, five random orders/state, one paired estimate/task. Negative and zero savings (open circles) are retained. Mean saving: 2.43 actions, task-paired bootstrap 95% CI $[0.82, 4.07]$. The sighting endpoint differs by 16 pp, CI $[-1, 34]$.

E.5.3 Source use and matched deployed-event continuations

All-task activation coverage. Table 3 inventories all three branches across the 134 tasks in the reference deployment. The 83 activations occur on 64 distinct tasks, with overlapping branch task sets. Direct-effect recovery accounts for 66 activations on 53 tasks; the matched study below analyzes all 17 information/chain events. This inventory establishes deployment coverage, and the controlled removals isolate local support utility.

Table 3: Branch activation coverage in the $106/134$ ALFWorld deployment. Task counts use the full 134-task population; successes are final task outcomes among branch-activated tasks, whose sets may overlap.

Branch	Activations	Tasks	Final task successes
Direct-effect	66	53	52
Information	15	15	12
Chain	2	2	2

The compiled evidence is exercised in deployment. Figure 7 traces compiled evidence to its use in the reference deployment of Table 2 ($106/134$ tasks, 3,346 actions). Retrieval-linked cards cover all tasks in five task families—$24/24$ place, $31/31$ clean, $21/21$ cool, $23/23$ heat, and $17/17$ two-object—whereas look-at-in-light has no recorded retrieval or recovery coverage and only $3/18$ successes. This alignment motivates the branch-controlled analysis below. The information/chain subset contains 17 events—15 information-recovery and 2 chain activations—of which 15 reach their environment-confirmed local endpoint, 13 within three actions, and 14 occur in ultimately successful tasks. Their support spans 186 source trajectories from 169 source tasks, forming 203 source–target task pairs, including 95 cross-family pairs.

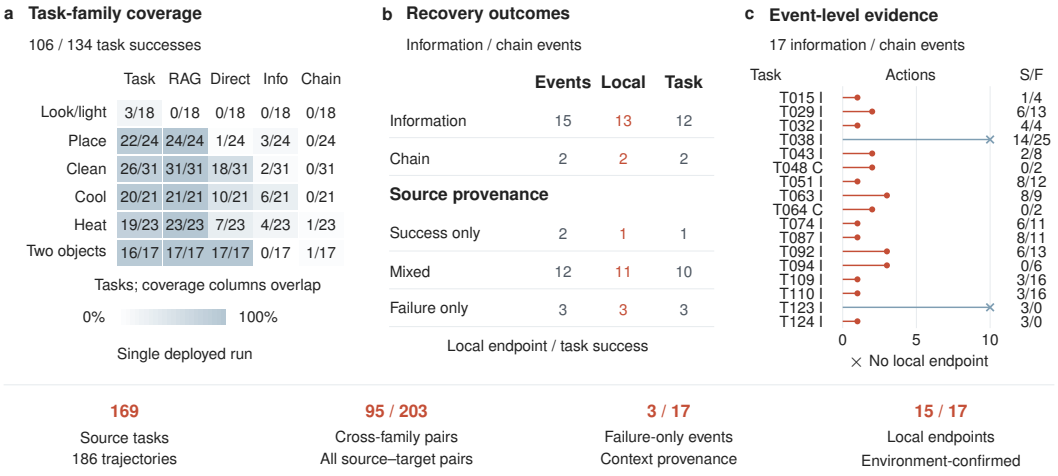

Figure 7: **Deployment linkage** (reference deployment; 106/134 successes). **(a)** Task-family success and overlapping mechanism coverage. **(b)** Events, local endpoints and task successes by mechanism/provenance, including all three failure-only events. Information endpoints are sightings; chain endpoints are pickups. **(c)** All 17 information/chain events: endpoint delay (dots), or control-window length without an endpoint (crosses); I/C: information/chain; S/F: successful/failed source episodes. Bottom: source-task breadth, cross-family pairs, failure-only context events, and observed local endpoints.

Relevant support improves matched deployed-event continuations. Matched forks test whether support relevance changes outcomes within the selected event cohort. Each of the 17 fixed events is continued under three arms—full support, removal of its relevant structured support, and equal-size removal of unrelated support—with pre-fork history, controller state, retrieval history, and the first policy proposal matched under the original 50-action budget (Figure 3c; detailed results in Figure 8). With full support, $15/17$ events reach their native local endpoint within five actions and $14/17$ tasks succeed; unrelated removal preserves both outcomes, whereas relevant removal drops them to $3/17$ and $5/17$. The effect spans both mechanisms: information recovery falls from $13/15$ local endpoints and $12/15$ task successes to $3/15$ and $5/15$, and both chain activations fall from $2/2$ to $0/2$. Mean restricted endpoint waiting time rises from 2.06 to 4.94 actions; nine tasks succeed only with full support and none only after removal. Capped completion cost rises from 32.47 to 44.35 actions per task, and continuation policy tokens more than double (842K to $1{,}796$K). All three events with failure-only context support succeed with full support; after relevant removal, two fail and the third requires 19 additional actions. Shared pre-intervention history makes this matched continuation design a direct test of support relevance.

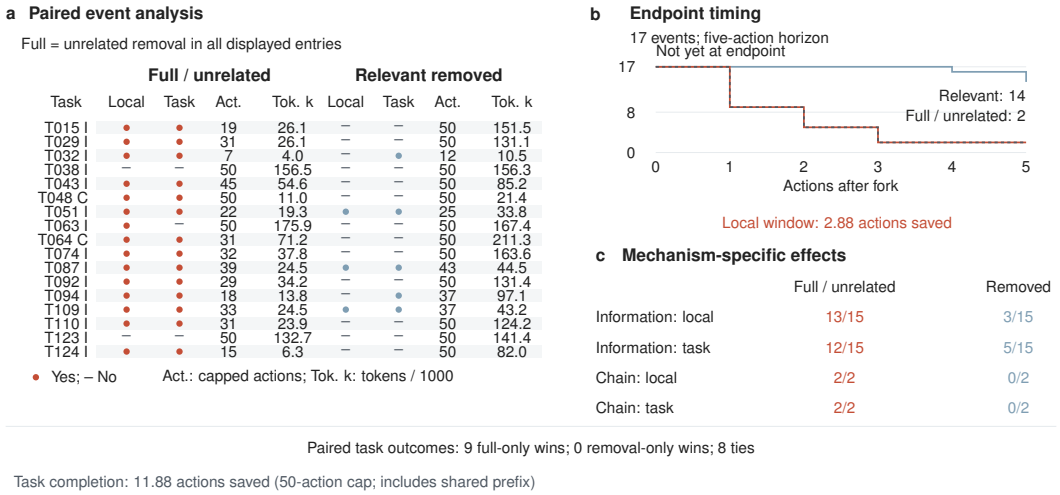

Figure 8: **Counterfactual analysis of 17 deployed information/chain events.** Three matched forks share native history, controller/retrieval state, and the first policy proposal under the original 50-action budget; unrelated removal is an equal-size control. **(a)** Five-action local endpoints, task success, capped completion actions, and continuation tokens (K). Full/unrelated columns coincide. I/C: information/chain; failed tasks cost 50 actions, and completion counts include the shared prefix. **(b)** Full support saves 2.88 actions within the local window and 11.88 actions under the task-completion cap. **(c)** Mechanism-specific outcomes, paired task wins, and ties. All arms share the continuation protocol and retrieval index.

Reading a success and a failure in the event comparison. For illustration, we select the lowest task index among full-only task wins and the lowest among full-support failures in the fixed 17-event cohort; Figure 8 retains every event. In the knife-cleaning case (T015), the fork follows action 12. Full support first sends the agent to countertop 2, where the knife becomes visible; the local endpoint is reached in one action and the task finishes at action 19. Relevant removal instead starts with inventory, misses the five-action endpoint, and fails at action 50. Equal-size unrelated removal matches full support. The matched class has one successful and four unsuccessful source episodes; this is mixed provenance, not failure-only evidence. In the mug-heating case (T038), full support visits countertop 3 but never reaches the local endpoint; both full and relevant-removal arms fail at action 50. Its 14 successful and 25 unsuccessful sources show why evidence support and online admission complement policy competence. For the knife-cleaning task, the matched full-support continuation finishes at action 19 and the native deployment at action 28.

E.5.4 Source breadth and outcome provenance

Unsuccessful episodes supply support that successful ones do not. The count of compiled context-support classes grows with pool size—39, 44, 58, 62, 69 classes at 100, 150, 300, 450, 600 source episodes—with new classes still appearing at the largest tier. At the full pool, 22 of 69 classes receive context support only from unsuccessful episodes, 43 from mixed outcomes, and 4 only from successful episodes. Excluding unsuccessful-source witnesses removes roughly a third of the catalog, demonstrating their distinct contribution to recovery coverage. Effect evidence $\mathcal{V}_c$ remains admitted separately (Section 3.2).

Source evolution of condition classes. Figure 9 separates breadth from volume: from 100 to 600 source episodes, candidate contracts grow $110 \rightarrow 684$, while condition classes grow $39 \rightarrow 69$ (a/b). The last tier adds seven classes, six with failure-only context provenance in the final pool (c). Additional sources therefore reinforce existing classes as well as expose previously absent contexts. Contract counts quantify context-source records, and panel d shows how provenance changes as sources accumulate.

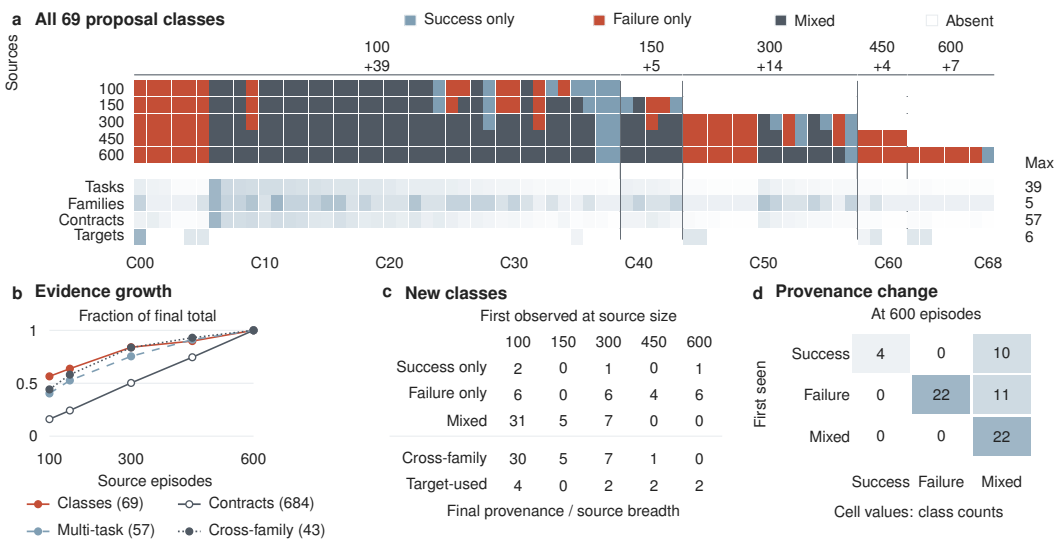

Figure 9: **Source evolution of local-condition proposals. (a)** All 69 classes across five separately rebuilt nested subsets; brackets mark first appearance/new-class counts. Bottom rows: final distinct tasks, families, contracts and target-cohort links, each normalized to its displayed maximum. **(b)** Growth normalized to final totals listed in the legend. **(c)** Final provenance, cross-family and target-use counts by discovery cohort. **(d)** First-appearance versus 600-episode provenance. Counts describe offline proposals before online admission.

E.5.5 Cross-task transfer and source redundancy

Transfer landscape over the fixed catalog. Figure 10 separates transfer reach from source redundancy. Of 72 source–target links, 42 cross families (b), while removing the largest source changes decisions in all 14 single-source states and none of eight multi-source states (d). Cross-family transfer and resilience to source loss are therefore distinct. The state-level analysis (a) retains escape and removal outcomes, while c shows source breadth across all 69 classes.

Reading the state-level transfer analysis. For a concrete contrast in Figure 10a, take the first displayed single-source and multi-source states. The keychain state has one failure-only source and escapes in two actions with full or unrelated-removed support, but not within 12 actions after relevant removal in any of three seeds; removing its sole source also changes the controller decision. The mug state has nine source tasks and ten contracts, all with unsuccessful-source context provenance. It also escapes in two actions with full support and fails after relevant removal, yet its decision survives largest-source removal. Thus collective relevance and resilience to losing one source are complementary properties.

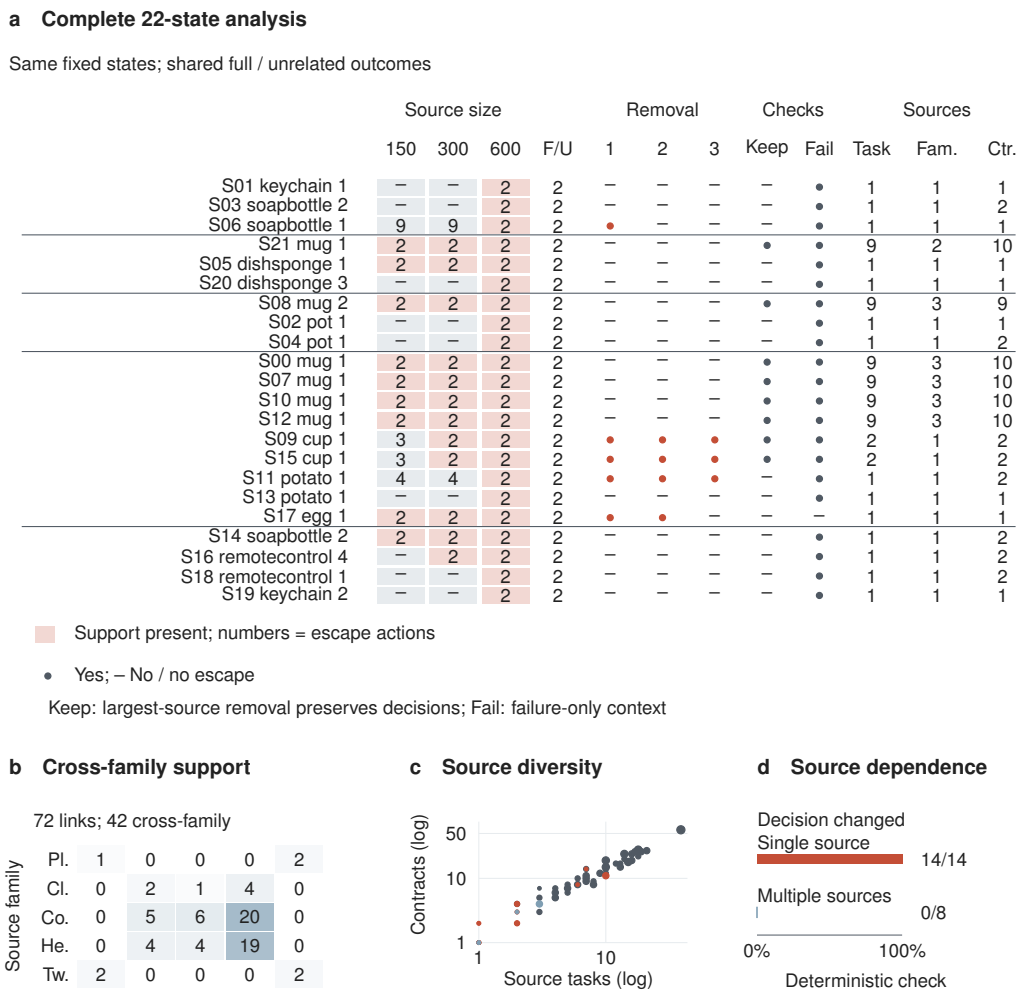

Figure 10: **Cross-task transfer landscape. (a)** All 22 states by family; S00–S21 are stable checkpoint IDs shared with Figure 11. Tint: support; numbers: escape actions; diamonds in removal columns: escape within the horizon; dash: no escape. F/U combines identical full/unrelated escape times; removal columns: three fixed relevant-removal seeds; Keep: decisions preserved under every tied largest-source removal; Fail: failure-only context support. Task/Fam./Ctr.: source-task/family/contract counts. **(b)** Pl./Cl./Co./He./Tw.: place, clean, cool, heat, two-object families; 42 of 72 links cross families. **(c)** All 69 classes on log task/contract axes; area: family breadth; vermilion/blue-gray/dark gray: failure-only/success-only/mixed. **(d)** Largest-source removal changes decisions in 14/14 single-source and 0/8 multi-source states, with all tied largest sources tested.

E.5.6 Checkpoint- and class-level retention

Detailed source-retention analysis. Figure 11 complements the aggregate curves and prediction check in Figure 4 with every checkpoint's and class's retention frequency. Each cell is the fraction of 200 outcome- and family-stratified nested draws retaining support at the indicated source-pool size. The source counts in panel a distinguish singly supported checkpoints from those with redundant sources; panel b exposes class-level heterogeneity hidden by aggregate coverage. Class IDs match Figure 9. Prediction assumes uniform sampling within strata conditional on quotas, approximated by the hashed orderings; full-pool retention is 100%.

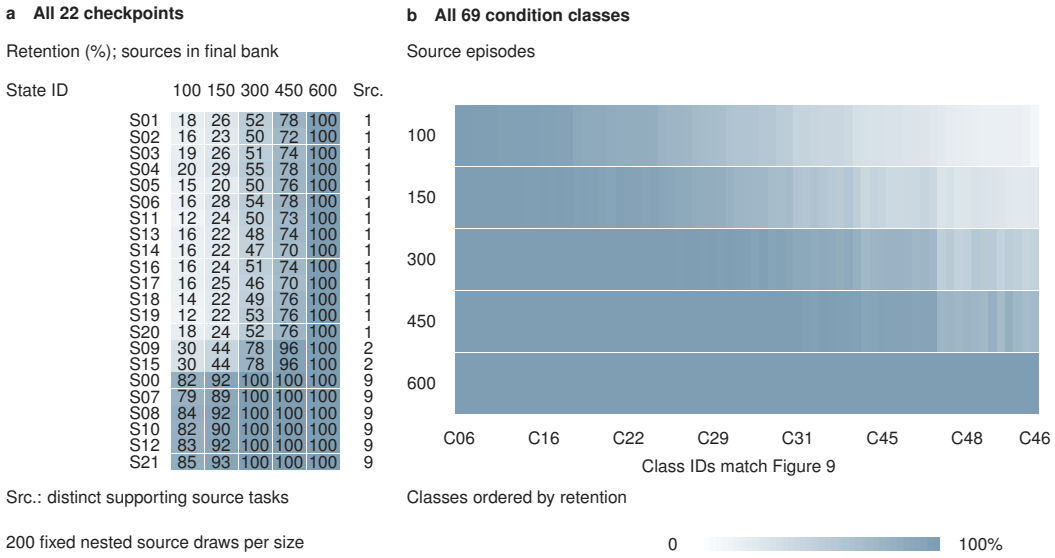

Figure 11: **Checkpoint- and class-level source retention. (a)** All 22 checkpoints (S00–S21, also used in Figure 10): retention percentages across five source sizes and distinct supporting source-task counts in the final bank. **(b)** All 69 classes ordered by retention, with IDs aligned to Figure 9. Both panels use the same 0–100% scale and the same 200 nested draws per size. Aggregate coverage, draw spread, and prediction agreement appear only in Figure 4.

E.6 Fixed-bank direct-effect intervention

Isolating online corrective execution. The three PMR mixed-pool evaluations in Table 8 are reused as the full arm here; they are the PMR-reference rows expanded in Table 10, not its separate retrieval-only arm. Each is paired with a direct-effect-off evaluation on the same 134 unseen ALFWorld tasks, using the same compiled structural support, cards, retrieval index, and task protocol within the pair. All controller switches are identical except direct-effect execution. This intervention therefore tests whether the online corrective branch contributes when the evidence available to the controller is held fixed.

Table 4: Fixed-bank direct-effect intervention on 134 unseen ALFWorld tasks. The PMR rows are the three mixed300 PMR-reference runs also reported in Tables 8 and 10; each is paired with a direct-effect-off run. The retrieval-only arm in Table 10 is a separate recovery-versus-retrieval baseline, not the fixed-bank control. Calls/task and tokens/task are normalized by 134 tasks; the direct-effect-off totals are 4,406/4,148/4,581 calls and 15.325/13.226/15.873M tokens.

Configuration	Run	Successes	SR (%)	Calls/task	Tok/task (K)	Gain (pp)
PMR	1	104	77.61	25.84	73.6	–
PMR, direct-effect off	1	81	60.45	32.88	114.4	+17.16
PMR	2	96	71.64	26.60	76.8	–
PMR, direct-effect off	2	79	58.96	30.96	98.7	+12.69
PMR	3	98	73.13	26.63	74.4	–
PMR, direct-effect off	3	84	62.69	34.19	118.5	+10.45
PMR, mean			74.13	26.36	74.9	
Direct-effect off, mean			60.70	32.67	110.5	+13.43

Across the three paired runs, enabling direct-effect execution raises mean success from 60.70% to 74.13%, a 13.43-point gain. It also reduces normalized model calls by 19.3% and online tokens by 32.2%. Every pair favors the full controller. Because the compiled evidence and retrieval index

are fixed within each pair, this comparison directly links the gain to the online corrective-execution channel rather than to a different memory construction.

F SUPPLEMENTARY RESULTS AND PER-RUN RECORDS

This appendix provides exact-value summaries followed by per-run outcomes and costs. Mechanism interpretations accompany their figures in Appendix E.5. ALFWorld component runs record one model call per environment step; other comparisons report calls separately. Repeated-run summaries use run-level metrics. Latencies report model-request wall clock under each study's serving configuration, while cross-environment comparisons use success, actions, calls, and tokens.

F.1 AGGREGATE COMPARISON AND EXTENDED EXPERIMENTAL EVIDENCE

The following summaries organize the study cohorts and complete per-run records supporting Section 4.

Table 5: Success rate (%), mean ± sample SD. Three runs per cell except ALFWorld unseen Synapse ($n = 4$). Task counts: 140/134/100/100. Per-run results: Appendix F.8. Method-specific construction and deployment scope: Appendix B.3.

Method	ALF. seen	ALF. unseen	TextCraft	BabyAI
P (policy-only)	42.4 ± 2.9	43.0 ± 1.6	61.7 ± 0.6	37.3 ± 2.1
Synapse	44.8 ± 2.2	45.0 ± 2.5	76.7 ± 1.5	39.7 ± 1.5
AWM	43.3 ± 2.3	37.1 ± 1.9	54.3 ± 4.7	32.3 ± 2.1
ReasoningBank	51.4 ± 2.6	46.8 ± 4.1	70.7 ± 2.5	30.0 ± 3.6
AdaMEM-LOW	56.2 ± 1.1	53.7 ± 3.3	65.3 ± 2.1	42.3 ± 3.5
AdaMEM-HIGH	58.3 ± 3.7	55.0 ± 0.4	71.3 ± 3.5	44.7 ± 3.5
PMR	$\mathbf{76.9 \pm 1.8}$	$\mathbf{77.6 \pm 1.3}$	$\mathbf{80.0 \pm 0.0}$	$\mathbf{58.0 \pm 1.0}$

Component comparison. Table 2 reports ALFWorld unseen success for the full method, its two branch ablations, the retrieval-only configuration, and the policy-only baseline, together with action cost; complete token accounting remains in Table 11. PMR ranks first in all three repeats (Table 11). Averaged over seeds, the full method solves $104.0/134$ tasks (77.6%) against $62.3/134$ (46.5%) for the policy-only baseline, a gain of 31.1 percentage points; the worst PMR run ($103/134$) exceeds the best run of every other arm. Retrieval alone moves the policy from 46.5% to 56.5%; adding condition-checked recovery raises success to 77.6%. These end-to-end configuration contrasts motivate separating retrieval from execution, while the fixed-bank paired intervention isolates the online direct-effect branch.

Token totals are rounded to 0.1 million; normalized token costs are approximate. Tokens/task (K), mean ± sample SD, are 115.7 ± 1.3 for P, 105.5 ± 1.1 for PM, 108.7 ± 3.4 without direct-effect recovery, 76.1 ± 0.7 without information recovery, and 74.9 ± 3.0 for PMR. The full method combines the highest success with 25.1 environment steps and 74.9K tokens per task, compared with 35.1 steps and 115.7K tokens for P. Every arm issues one model call per submitted step and no auxiliary recovery-generation calls, directly linking shorter successful trajectories to lower online cost.

Cross-method comparison on TextCraft. Table 6 tests whether the benefit extends beyond the primary policy. PMR reaches 40.3% with Llama-3.2-3B and 37.3% with Qwen2.5-3B, exceeding the best reference including P by 15.0 and 13.7 points, respectively. The runner-up is AdaMEM-LOW for Llama and P for Qwen2.5; among reuse baselines alone, Synapse leads on Qwen2.5. PMR ranks first in each of the six rounds, and its lowest recorded success ($36/100$) exceeds the highest reference run ($27/100$). This result extends PMR's advantage across two additional frozen policies.

Reading the two axes together. Figure 12a–b plots TextCraft success against online token cost under two frozen policy models. Higher-left positions indicate higher success at lower token cost.

Table 6: TextCraft (100 tasks) under two frozen policy models: mean ± standard deviation over three rounds for success rate, episode length, and token cost (per-round values in Appendix F; auxiliary-call and latency profile in Table 7). **Bold** best, underline second best. Our method is the only one that improves all three columns over the policy-only baseline; every competing mechanism costs more tokens than no reuse at all. Row order is fixed and not sorted by score.

Method	Llama-3.2-3B			Qwen2.5-3B		
	SR (%)	Steps	Tok/task (K)	SR (%)	Steps	Tok/task (K)
P (policy-only)	23.0 ± 2.6	24.2 ± 0.4	24.3 ± 0.2	23.7 ± 1.5	23.9 ± 0.5	23.9 ± 0.7
Synapse	23.7 ± 0.6	25.0 ± 0.3	46.9 ± 6.3	22.7 ± 3.2	24.8 ± 0.7	42.5 ± 1.6
AWM	22.3 ± 2.1	25.0 ± 0.2	98.8 ± 30.7	21.0 ± 2.6	24.8 ± 0.6	55.3 ± 1.0
ReasoningBank	20.0 ± 1.0	25.3 ± 0.2	31.0 ± 5.0	22.3 ± 0.6	24.9 ± 0.3	27.9 ± 0.2
AdaMEM-LOW	25.3 ± 0.6	24.3 ± 0.1	71.2 ± 11.5	19.0 ± 2.6	25.5 ± 0.5	67.1 ± 1.1
AdaMEM-HIGH	24.7 ± 2.5	24.2 ± 0.4	105.3 ± 6.1	20.3 ± 0.6	24.9 ± 0.1	102.4 ± 0.9
PMR (full)	**40.3 ± 4.2**	**21.6 ± 0.3**	**22.5 ± 0.3**	**37.3 ± 1.5**	**22.1 ± 0.7**	**21.7 ± 0.9**

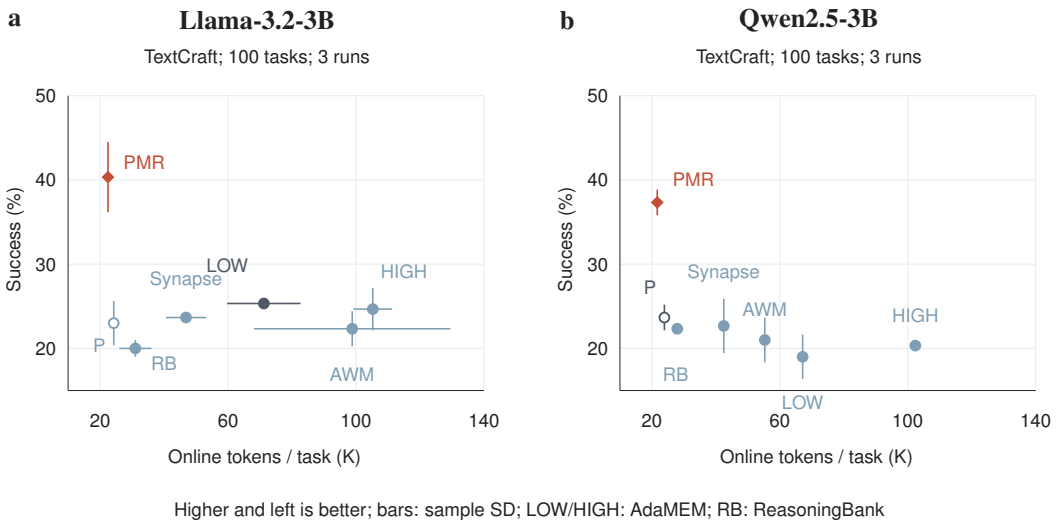

Figure 12: **Cross-model performance and cost.** TextCraft success and online tokens/task (K) under two frozen policies; token SDs are in Table 6. Points show three-run means; horizontal/vertical intervals show sample SD for tokens/success. Common linear axes. Red diamond: PMR; dark gray: highest-success comparator in each panel; open circle: policy-only. Per-run values: Appendix F.

PMR occupies that region relative to all evaluated comparators in both panels; the horizontal and vertical intervals retain the three-run variability. Table 6 remains the exact-value reference. AdaMEM-LOW/HIGH jointly vary refresh decisions and generated strategies; Table 7 reports their realized system-level profiles.

F.1.1 Cost profile and cross-model behavior

The online loop adds no auxiliary generative requests for recovery (Appendix C.3); the competing strategy-synthesis variants add requests according to their realized refresh schedules. Table 7 reports token intensity, strategy-update frequency, and latency on the runs of Table 6. Total token savings are empirical, since prompt size and episode length can offset a lower call count. The auxiliary term behaves as predicted. Our strategy-update rate is exactly 0.0% in every round on both policy models: all candidate generation is offline, and $\mathrm{Resolve}_{\Phi}$ is a symbolic evaluation over Φ, so no model call is issued that does not produce a candidate action. The measured AdaMEM rates are 6.8% and 8.6% of steps in the LOW configuration and 82.2% and 86.2% in the HIGH configuration, confirming that the two configurations differ as intended.

Table 7: TextCraft cost profile on the runs of Table 6, mean ± standard deviation over three rounds. *Upd* is the measured fraction of steps with an auxiliary strategy-synthesis call; it is zero for methods without such an online synthesis branch (including PMR), and nonzero when those requests are issued (Appendix C.3). Latency (s/task) is model-request wall clock under the shared serving configuration. **Bold** best, underline second best.

	Llama-3.2-3B			Qwen2.5-3B		
Method	Tok/step	Upd (%)	s/task	Tok/step	Upd (%)	s/task
P (policy-only)	**1003**	0.0	57.4 ± 17.5	1002	0.0	40.1 ± 2.6
Synapse	1872	0.0	133.5 ± 125.4	1712	0.0	49.5 ± 3.4
AWM	3953	0.0	139.4 ± 13.8	2228	0.0	49.4 ± 9.2
ReasoningBank	1227	0.0	125.1 ± 102.6	1120	0.0	51.2 ± 3.3
AdaMEM-LOW	2927	6.8 ± 1.8	312.7 ± 253.2	2634	8.6 ± 0.6	138.9 ± 5.7
AdaMEM-HIGH	4346	82.2 ± 5.1	427.1 ± 225.9	4113	86.2 ± 0.6	236.4 ± 13.8
PMR (full)	1038	0.0	**55.1 ± 18.1**	**979**	0.0	**34.5 ± 0.7**

PMR uses 22.5K and 21.7K tokens per task, compared with 105.3K and 102.4K for AdaMEM-HIGH in the two cross-model cohorts. Table 7 separates total input-plus-output tokens per environment step from the frequency of auxiliary strategy synthesis. PMR combines lower token intensity than the reuse baselines with shorter mean episodes (Table 6). Recorded request latency also favors PMR under the shared serving configuration.

Secondary-environment source-pool records. Table 12 reports all six success-only-pool runs on TextCraft and BabyAI, including recovery authorization and cost. Together with the ALFWorld source-composition study, these cohorts show how PMR operates across source regimes and environments. Both issue zero auxiliary recovery-generation calls; authorization counts accepted interventions while the resolver is evaluated at every step (Algorithm 2).

F.1.2 Component analysis

The two branch ablations in Table 2 show unequal configuration-level changes for the branches of Section 3.4. The direct-effect-disabled configuration solves 30.3 fewer tasks on average (22.6 percentage points) than the full reference, with lower success in all three repeats (-29, -33, -29). Its mean (55.0%) is close to retrieval alone (56.5%). The information-disabled configuration solves 5.0 fewer tasks on average (3.7 points), with differences of -8, -5, and -2 tasks. The asymmetry is consistent with their roles: direct effects can intervene under supported current conditions before a new stall develops, whereas information recovery fires on the narrower trigger of a repeated action with no observed event. These end-to-end ablations vary the branch together with its dependent artifacts. The fixed-bank paired ablation complements them by keeping compiled evidence and retrieval fixed while disabling only online direct-effect execution.

Supplementary representation and retrieval probes. Three further ablations—presenting raw source segments in place of rendered cards, triggering discovery on repeated actions in place of the full impasse condition, and removing the goal-compatibility filter from retrieval—remain within one to two ALFWorld tasks of the full method, including one task-for-task match. The result demonstrates robustness to card rendering, impasse detection, and compatibility-filter variants while preserving PMR's central evidence-to-recovery design.

F.2 Source-composition evaluations

Table 8 reports repeated source-pool evaluations on 134 unseen ALFWorld tasks. The PMR block compares a 300-episode mixed pool (150 successful and 150 unsuccessful episodes) with a 300-success pool. The three PMR mixed-pool rows are reused as the full arm of the fixed-bank intervention in Table 4; that table adds the paired direct-effect-off runs. The AdaMEM block applies the same mixed-versus-success-only source comparison to LOW and HIGH controller configurations. These records complement the source-provenance analysis in Figure 3.

Table 8: Source-pool comparisons: per-run successes out of 134 and success rates. S/F denote successful/unsuccessful source episodes. The PMR rows are the matched source-pool cohort; the AdaMEM rows compare the same pool compositions under LOW and HIGH controller configurations.

Method	Source configuration	Successes	SR (%)
PMR source-pool cohort			
PMR	150 S + 150 F	104 / 96 / 98	77.61 / 71.64 / 73.13
PMR	300 S	88 / 91 / 90	65.67 / 67.91 / 67.16
AdaMEM source-composition cohort			
AdaMEM-LOW	150 S + 150 F	75 / 76 / 77	55.97 / 56.72 / 57.46
AdaMEM-LOW	300 S	81 / 79 / 77	60.45 / 58.96 / 57.46
AdaMEM-HIGH	150 S + 150 F	69 / 76 / 72	51.49 / 56.72 / 53.73
AdaMEM-HIGH	300 S	75 / 74 / 74	55.97 / 55.22 / 55.22

Within the PMR source-pool cohort, the mixed and success-only pools yield mean success rates of 74.1% and 66.9%, respectively. Across the AdaMEM runs, the LOW mixed and success-only pools yield 56.72% and 58.96%, while the HIGH mixed and success-only pools yield 53.98% and 55.47%. The mixed pool is held at 150 successful plus 150 unsuccessful episodes for both controller settings; the primary PMR comparison uses its designated 77.6% cohort. The fixed-bank table uses the same three PMR mixed-pool rows to isolate the online direct-effect branch.

F.3 TextCraft: repeated runs across two policies

Table 9: TextCraft, 100 tasks: complete metrics for every method and round. *Upd* is the fraction of steps with an auxiliary strategy-synthesis call; it is zero for methods without such an online synthesis branch (including PMR), and nonzero when those requests are issued.

Policy / round	Method	Succ.	SR%	Steps	Tok/task (K)	Tok/step	Upd%	s/task
Llama-3.2-3B, run 1	PMR	45	45.0	21.4	22.6	1,058	0.0	76.0
	P	26	26.0	23.7	24.3	1,024	0.0	77.6
	AWM	20	20.0	25.2	63.4	2,517	0.0	123.7
	Synapse	23	23.0	25.3	54.1	2,143	0.0	278.3
	ReasoningBank	20	20.0	25.3	36.8	1,456	0.0	243.6
	AdaMEM-LOW	26	26.0	24.4	84.2	3,457	4.7	605.1
	AdaMEM-HIGH	25	25.0	23.9	111.6	4,660	76.5	687.8
Llama-3.2-3B, run 2	PMR	39	39.0	21.9	22.1	1,005	0.0	44.1
	P	21	21.0	24.5	24.4	994	0.0	46.7
	AWM	24	24.0	24.9	115.3	4,631	0.0	145.0
	Synapse	24	24.0	24.7	42.6	1,723	0.0	59.7
	ReasoningBank	19	19.0	25.2	27.9	1,111	0.0	64.9
	AdaMEM-LOW	25	25.0	24.2	62.6	2,585	8.1	161.9
	AdaMEM-HIGH	22	22.0	24.6	99.5	4,052	86.2	288.6
Llama-3.2-3B, run 3	PMR	37	37.0	21.6	22.7	1,051	0.0	45.2
	P	22	22.0	24.3	24.1	992	0.0	47.9
	AWM	23	23.0	25.0	117.8	4,712	0.0	149.6
	Synapse	24	24.0	25.1	43.9	1,749	0.0	62.4
	ReasoningBank	21	21.0	25.5	28.4	1,114	0.0	66.9
	AdaMEM-LOW	25	25.0	24.4	66.8	2,738	7.6	171.2
	AdaMEM-HIGH	27	27.0	24.2	104.7	4,326	83.9	304.8
Qwen2.5-3B, run 1	PMR	39	39.0	21.5	20.8	968	0.0	33.7
	P	22	22.0	24.4	24.6	1,008	0.0	43.0
	AWM	18	18.0	25.4	56.1	2,209	0.0	59.9
	Synapse	19	19.0	25.6	44.1	1,724	0.0	53.3
	ReasoningBank	22	22.0	24.7	27.8	1,125	0.0	54.9
	AdaMEM-LOW	16	16.0	26.0	67.4	2,595	7.9	145.2
	AdaMEM-HIGH	20	20.0	24.9	102.5	4,124	86.6	251.7

Continued on next page

Table 9 (continued)

Policy / round	Method	Succ.	SR%	Steps	Tok/task (K)	Tok/step	Upd%	s/task
Qwen2.5-3B, run 2	PMR	36	36.0	22.8	22.5	988	0.0	34.8
	P	25	25.0	23.5	23.3	993	0.0	38.0
	AWM	23	23.0	24.3	54.2	2,232	0.0	43.0
	Synapse	25	25.0	24.3	41.0	1,691	0.0	46.6
	ReasoningBank	22	22.0	25.2	27.9	1,107	0.0	48.8
	AdaMEM-LOW	21	21.0	25.1	65.9	2,625	9.1	134.0
	AdaMEM-HIGH	20	20.0	25.0	101.4	4,053	86.4	225.0
Qwen2.5-3B, run 3	PMR	37	37.0	22.1	21.7	982	0.0	35.1
	P	24	24.0	23.7	23.8	1,004	0.0	39.4
	AWM	22	22.0	24.8	55.6	2,242	0.0	45.3
	Synapse	24	24.0	24.6	42.3	1,720	0.0	48.7
	ReasoningBank	23	23.0	24.9	28.1	1,129	0.0	49.9
	AdaMEM-LOW	20	20.0	25.4	68.1	2,681	8.7	137.6
	AdaMEM-HIGH	21	21.0	24.8	103.2	4,161	85.5	232.4

F.4 ALFWORLD: MECHANISM-ANALYSIS COHORT

The recovery-enabled reference in this separate mechanism-analysis cohort solves 104, 96, and 98 of 134 tasks, giving $74.1\% \pm 3.1$ (mean $\pm$ sample SD). Table 8 reports the separate source-pool comparison, including the corresponding $300\,S$ PMR runs (88, 91, and 90 successes; 66.9% mean). Table 10 focuses on the mixed300 recovery-versus-retrieval contrast: it expands the same three $150\,S + 150\,F$ PMR-reference runs with per-run cost and step diagnostics and places them beside a retrieval-only arm built from the same mixed300 pool with recovery disabled. The success-only PMR rows remain in Table 8 and are not repeated here. These three mixed-pool PMR runs are also the full arm of the fixed-bank direct-effect intervention in Table 4, where they are paired with 81/79/84 direct-effect-off successes. The 77.6% cohort shared by the main comparison and component study remains the primary PMR result. The retrieval-only arm is reported only in this mechanism cohort. The $s1$–$s3$ labels here are local repeat labels and are not paired with the labels in Table 11.

Table 10: ALFWorld unseen (134 tasks, $B = 50$): separate mixed300 mechanism-analysis cohort. The three PMR-reference rows use the $150\,S + 150\,F$ pool and are the same runs summarized in Table 8; this table expands them with per-run diagnostics beside a retrieval-only arm using the same pool with recovery disabled. The success-only PMR source-pool rows are reported in Table 8, not repeated here. None of these rows is the component-study cohort in Table 11. Steps (solved) conditions on success; Cap denotes action-limit attainment. Tokens are per task; each run records one policy call per step and zero auxiliary calls.

Configuration	Record	Succ.	SR%	Steps	Steps (solved)	Cap%	Tok/task (K)	Tok/ step	p-tok/ task	c-tok/ task	s/task
Retrieval-only (mixed300)	s1	57	42.5	36.33	17.86	57.5	114.7	3,157	89,742	24,955	442.0
	s2	66	49.3	35.01	19.58	50.7	122.1	3,487	97,838	24,273	440.6
	s3	63	47.0	35.38	18.91	53.0	123.6	3,493	98,481	25,119	456.1
PMR reference (mixed300)	s1	104	77.6	25.84	18.88	23.1	73.6	2,849	56,374	17,263	326.8
	s2	96	71.6	26.60	17.33	28.4	76.8	2,886	59,347	17,417	334.5
	s3	98	73.1	26.63	18.04	26.9	74.4	2,794	57,218	17,182	339.4

F.5 ALFWORLD: RECOVERY-COMPONENT ABLATIONS

Table 11: ALFWorld unseen (134 tasks), component-study runs of the full method and its ablations. Token totals are rounded to 0.1 million; steps equal model calls in every run. The three full-method runs also support the main comparison. These rows measure end-to-end configurations with their dependent artifacts; the three-run fixed-bank intervention in Table 4 isolates online direct-effect execution.

Arm	Seed	Succ.	SR%	Total steps	Total calls	Tok (M)
PMR (full)	s1	106	79.1	3,346	3,346	10.1
	s2	103	76.9	3,339	3,339	9.6
	s3	103	76.9	3,408	3,408	10.4
PMR, no information branch	s1	98	73.1	3,518	3,518	10.1
	s2	98	73.1	3,493	3,493	10.3
	s3	101	75.4	3,455	3,455	10.2
PMR, no direct-effect branch	s1	77	57.5	4,204	4,204	14.1
	s2	70	52.2	4,491	4,491	15.0
	s3	74	55.2	4,327	4,327	14.6
PM (retrieval only)	s1	72	53.7	4,498	4,498	14.3
	s2	80	59.7	4,356	4,356	14.1
	s3	75	56.0	4,421	4,421	14.0
P (policy-only)	s1	65	48.5	4,647	4,647	15.3
	s2	59	44.0	4,780	4,780	15.6
	s3	63	47.0	4,692	4,692	15.6

F.6 TEXTCRAFT AND BABYAI: SUCCESS-ONLY EXPERIENCE POOLS

Table 12: TextCraft and BabyAI with the primary policy (Qwen3-4B), success-only artifacts, 100 tasks per seed: complete per-run metrics. Auth is the fraction of steps on which condition-checked recovery was authorized; auxiliary calls are zero throughout.

Environment	Seed	Succ.	SR%	Steps	Tok/task (K)	Tok/step	Auth%	s/task	s/step
TextCraft	s0	76	76.0	13.86	16.4	1,184	5.5	114.79	8.280
	s1	73	73.0	14.22	16.9	1,188	5.1	119.45	8.400
	s2	78	78.0	13.54	16.1	1,189	5.8	111.36	8.220
BabyAI	s0	56	56.0	72.66	51.8	713	0.7	13.38	0.184
	s1	53	53.0	74.12	53.4	721	0.9	14.16	0.191
	s2	58	58.0	70.84	50.2	708	0.6	12.82	0.181

F.7 TEXTCRAFT AND BABYAI: RECORDED EXECUTION COUNTS

Execution counters are reported for the first listed Qwen3-4B run in each secondary environment (s0); per-task rates in parentheses divide by 100 tasks. Overrides count policy proposals replaced by an admitted corrective action; authorized counts steps on which recovery was authorized.

Table 13: Execution accounting for the first listed Qwen3-4B run in each secondary environment (s0; 100 tasks each).

Counter	TextCraft	BabyAI
Success / planned	76 / 100	56 / 100
Total environment steps	1,386	7,266
Total model calls	1,386	7,266
Prompt tokens	1,160,469	5,113,923
Completion tokens	479,936	65,294
Total tokens	1,640,405	5,179,217
Model seconds	11,479.4	1,338.0
Authorized recovery steps	76	49
Overrides	68 (0.68/task)	32 (0.32/task)
Truncated outputs	13 (0.13/task)	0
Embedding calls	143 (1.43/task)	92 (0.92/task)
Embedding seconds	30.9	2.5

F.8 COMPLETE REPEATED-RUN COMPARISON

The four tables report 85 runs: seven methods on four fixed splits, with three runs per method except ALFWorld unseen Synapse ($n = 4$). Repeat numbers index runs within each method. The three unseen PMR runs also provide the full-method reference for the component study (Table 11). The tables use a common set of outcome and cost metrics within each split. Costs include failed episodes and replaced policy proposals, and exclude offline construction.

Table 14: ALFWorld seen: repeated-run outcomes and online cost. 140 tasks per run; action budget 50. Tokens are in thousands; all costs are per task.

Method	Repeat	SR (%)	Actions/task	Calls/task	Tokens/task (K)
P	1	39.29	37.94	37.94	115.45
P	2	45.00	36.61	36.61	106.80
P	3	42.86	36.99	36.99	106.57
Synapse	1	44.29	33.73	33.73	162.42
Synapse	2	47.14	33.41	33.41	160.15
Synapse	3	42.86	35.08	35.08	178.19
AWM	1	40.71	36.10	36.10	196.01
AWM	2	44.29	35.46	35.46	195.77
AWM	3	45.00	35.35	35.35	194.47
ReasoningBank	1	50.71	34.90	34.90	109.77
ReasoningBank	2	49.29	36.06	36.06	109.91
ReasoningBank	3	54.29	33.83	33.83	104.93
AdaMEM-LOW	1	55.00	31.83	68.56	205.93
AdaMEM-LOW	2	56.43	30.84	66.79	200.95
AdaMEM-LOW	3	57.14	30.60	66.41	201.11
AdaMEM-HIGH	1	59.29	31.36	57.61	246.83
AdaMEM-HIGH	2	54.29	31.94	60.57	259.63
AdaMEM-HIGH	3	61.43	31.74	56.21	223.51
PMR	1	77.14	25.66	25.66	64.50
PMR	2	78.57	24.62	24.62	57.30
PMR	3	75.00	26.02	26.02	66.69

Table 15: ALFWorld unseen: repeated-run outcomes and online cost. 134 tasks per run; action budget 50. Tokens are in thousands; all costs are per task. PMR token costs are approximate, normalized from totals rounded to 0.1M.

Method	Repeat	SR (%)	Actions/task	Calls/task	Tokens/task (K)
P	1	44.78	36.96	36.96	120.77
P	2	42.54	37.09	37.09	115.41
P	3	41.79	37.04	37.04	118.40
Synapse	1	41.79	36.90	36.90	195.90
Synapse	2	47.01	34.12	34.12	180.52
Synapse	3	47.01	34.01	34.01	173.36
Synapse	4	44.03	35.45	35.45	190.41
AWM	1	37.31	37.63	37.63	212.63
AWM	2	35.07	38.93	38.93	231.93
AWM	3	38.81	37.82	37.82	213.23
ReasoningBank	1	51.49	34.38	34.38	121.44
ReasoningBank	2	44.78	36.20	36.20	138.50
ReasoningBank	3	44.03	36.69	36.69	132.67
AdaMEM-LOW	1	51.49	32.47	71.33	238.19
AdaMEM-LOW	2	57.46	31.41	68.40	225.73
AdaMEM-LOW	3	52.24	31.79	68.72	223.06
AdaMEM-HIGH	1	54.48	32.91	63.63	266.49
AdaMEM-HIGH	2	55.22	30.95	59.86	257.09
AdaMEM-HIGH	3	55.22	32.83	66.04	292.44
PMR	1	79.10	24.97	24.97	≈ 75.4
PMR	2	76.87	24.92	24.92	≈ 71.6
PMR	3	76.87	25.43	25.43	≈ 77.6

Table 16: TextCraft: repeated-run outcomes and online cost. 100 tasks per run; action budget 30. Tokens are in thousands; all costs are per task.

Method	Repeat	SR (%)	Actions/task	Calls/task	Tokens/task (K)
P	1	61.00	16.88	16.88	20.79
P	2	62.00	16.38	16.38	21.69
P	3	62.00	16.19	16.19	19.22
Synapse	1	77.00	13.82	13.82	29.17
Synapse	2	75.00	13.57	13.57	28.48
Synapse	3	78.00	13.31	13.31	28.67
AWM	1	56.00	19.46	19.46	229.95
AWM	2	58.00	18.67	18.67	219.98
AWM	3	49.00	19.70	19.70	232.40
ReasoningBank	1	68.00	15.48	15.48	23.79
ReasoningBank	2	71.00	15.40	15.40	23.18
ReasoningBank	3	73.00	14.78	14.78	22.15
AdaMEM-LOW	1	67.00	14.56	31.52	57.59
AdaMEM-LOW	2	66.00	15.29	33.16	65.44
AdaMEM-LOW	3	63.00	15.77	34.44	66.62
AdaMEM-HIGH	1	75.00	13.99	17.39	37.93
AdaMEM-HIGH	2	71.00	14.26	18.24	48.76
AdaMEM-HIGH	3	68.00	15.09	19.45	42.23
PMR	1	80.00	13.80	13.80	25.46
PMR	2	80.00	13.33	13.33	18.58
PMR	3	80.00	13.28	13.28	24.89

Table 17: BabyAI: repeated-run outcomes and online cost. 100 tasks per run; action budget 128. Tokens are in thousands; all costs are per task.

Method	Repeat	SR (%)	Actions/task	Calls/task	Tokens/task (K)
P	1	35.00	92.83	92.83	65.72
P	2	39.00	89.78	89.78	63.83
P	3	38.00	89.50	89.50	63.87
Synapse	1	38.00	83.70	83.70	212.04
Synapse	2	40.00	84.39	84.39	227.51
Synapse	3	41.00	79.12	79.12	205.17
AWM	1	33.00	84.92	84.92	286.73
AWM	2	34.00	82.68	82.68	278.79
AWM	3	30.00	85.47	85.47	288.22
ReasoningBank	1	34.00	86.11	86.11	83.86
ReasoningBank	2	27.00	90.03	90.03	87.67
ReasoningBank	3	29.00	87.48	87.48	85.00
AdaMEM-LOW	1	39.00	78.50	162.34	235.77
AdaMEM-LOW	2	46.00	74.16	152.94	221.88
AdaMEM-LOW	3	42.00	76.68	158.02	228.49
AdaMEM-HIGH	1	41.00	77.98	131.12	238.75
AdaMEM-HIGH	2	48.00	74.22	124.30	225.12
AdaMEM-HIGH	3	45.00	74.92	126.86	230.37
PMR	1	58.00	73.31	73.31	52.31
PMR	2	57.00	74.22	74.22	53.14
PMR	3	59.00	72.96	72.96	51.99