RE-GROUNDING RELATIONAL EXPERIENCE UNDER CHANGING AUTHORITY

Anonymous authors
Paper under double-blind review

ABSTRACT

An agent can learn a useful workflow from other people without acquiring their authority to execute it. When its role or authority changes, which parts should it reuse? We introduce REGROUND, a system that re-grounds past experience in the acting agent's current position. Its main algorithm constructs a reasoned realization gap: unmet requirements, their blockers, and constraints on completion. This gap guides retrieval, reuse, and search for a permitted way to achieve the original goal. Directed memory preserves experience, while a second algorithm extracts candidate authority changes from language and checks them against supporting evidence. On a synthetic planning benchmark, our planner uses 48.2% fewer transition attempts than Strong Repair on 216 feasible cases, with equal completion. Across all 432 cases, however, it uses 2.72 times as many attempts. On the supplied single-goal benchmark, our planner finishes every feasible case in its first retrieved slice; repeated unsuccessful searches accumulate cost on external-dependent cases. In a multigoal diagnostic, unmet-requirement retrieval reduces search relative to goal-only retrieval, although Strong Repair completes more tasks. In the tested API configuration, normative grounding improves permission-query accuracy from 90/106 to 103/106 at higher token cost, but performs worse on the tested local model. Unified and separated memory implementations with the same typed semantics tie; role-conditioned prompting still misapplies experience. The results show conditional search savings and failures in search allocation and language interpretation.

1 INTRODUCTION

An assistant observes a researcher explain a failed experiment, compare alternatives, and ask a principal investigator for a decision. It learns the reporting practice even though it was not the speaker. Later, the assistant is asked to prepare a similar report, but its permission to send reports directly has been revoked. The analysis may still be useful, and the report is still due. The assistant must find a permitted delivery route or keep the obligation pending.

Experience memory and plan repair address parts of this problem. Learned memories recover strategies and workflows from previous interactions (Ouyang et al., 2026; Wang et al., 2024; Shinn et al., 2023). Repair methods adapt plans when their conditions change (Nebel & Koehler, 1995; Fox et al., 2006). In a collaborative setting, reuse also depends on who acts. A practice learned from one role may fit another role's responsibilities yet contain steps that the second role cannot perform. Observing a successful action provides experience, not permission. Deleting all memory after a permission change would also discard useful knowledge.

Past experience may remain useful after the authority under which it was realized has changed. REGROUND represents why the old realization fails and uses that information to search for a new permitted realization of the same goal. Its main algorithm replays the historical workflow under the agent's current role, state, and authority. Unmet requirements, their blockers, and completion constraints form a linked gap $Q_t = (U_t, B_t, C_t)$. The gap guides experience retrieval and action admission. Search combines reusable steps with additional actions, widens the candidate set, and releases historical ordering when the initial fragments do not work.

Two components support this algorithm. Natural-language normative grounding (C) extracts candidate changes from conversation and checks the evidence that gives them institutional force. Dual

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.

relational memory (A) records directed collaboration experience separately from the resulting normative state. The agent can learn from any interaction it is permitted to observe, then assess that experience from its own current position. This is the principle *learn globally, act positionally*. The separation concerns update rules and meaning; one typed store can implement it.

Our principal contribution is the linked-gap interface from changes in role and authority to experience retrieval, action admission, and completion search (B). C supplies the language-to-authority update procedure; A supplies the memory architecture. The search heuristic, plan repair, and access-control machinery are established mechanisms. We evaluate the resulting system and find useful search guidance on one feasible task family, higher cost elsewhere, and strong dependence on language interpretation.

2 Related work

Learning reusable experience. ReasoningBank distills strategies from successful and failed trajectories, while Agent Workflow Memory extracts reusable workflows (Ouyang et al., 2026; Wang et al., 2024). Reflexion and ReAct connect feedback, reasoning, and action (Shinn et al., 2023; Yao et al., 2022). Persistent memory systems organize retained information in different forms (Packer et al., 2023; Zhong et al., 2023; Chhikara et al., 2025; Xu et al., 2025). G-Memory already conditions retrieved experience on agent roles (Zhang et al., 2025). Our question is what remains reusable after the acting participant's authority changes; role conditioning alone is not the contribution.

Memory and authority. AuthMem-Bench studies authority collapse during consolidation, and PPMF preserves provenance for tool authorization (Zhan et al., 2026; Xu et al., 2026). EAL-Bench tracks evolving authorization histories and tests source checking and event-sourced state updates (Cerruti et al., 2026). GateMem evaluates access and utility in multi-principal shared memory (Ren et al., 2026). These are direct predecessors for A/C's authority boundaries. We connect authority impact to experience-guided completion of the original goal, rather than claiming that memory-aware authorization is new.

Repair and authority. Plan repair studies the cost and stability of adapting earlier solutions (Nebel & Koehler, 1995; Fox et al., 2006). PlanFence validates action-relevant record dependencies, then replans or blocks after revisions (Chen et al., 2026). We use standard heuristic search (Bonet & Geffner, 2001). The tested addition is a linked gap that guides experience selection and action admission. Role- and attribute-based access control describe permissions and contextual restrictions (Sandhu et al., 1996; Hu et al., 2014). Trust management and truth maintenance track delegation and supporting assumptions (Li et al., 2002; Doyle, 1979; de Kleer, 1986); these inform C's treatment of revocation and independent supports. We do not benchmark against the recent systems above or claim superiority to them.

Evaluation scope. LongMemEval and LoCoMo test extended interaction (Wu et al., 2024; Maharana et al., 2024); interactive benchmarks test realistic agent tasks (Zhou et al., 2023; Koh et al., 2024; Xie et al., 2024; Drouin et al., 2024). Security studies examine attacks, trust boundaries, and memory-driven drift (Debenedetti et al., 2024; Zhan et al., 2024; Chen et al., 2024; Li & Zhu, 2026). Our finite symbolic worlds permit independent replay of actions and state updates. They do not establish long-history competence or adversarial robustness.

3 Problem and system

At time t, agent A has goal g, state s_t, action catalog $\mathcal{A}$, role position $p_t(A)$, and normative graph G_t^{RN}. Actions specify actors, preconditions, effects, protected operations, and scope/provenance restrictions. Historical workflow π^h achieved g under earlier conditions. The task is to achieve g under current conditions, retaining unmet requirements as pending when external change is needed.

Figure 1 separates what the agent has learned from what it is currently authorized to do; Figure 2 shows how ReGround reconciles the two during reuse. Experience can persist across authority changes without carrying its former authorization.

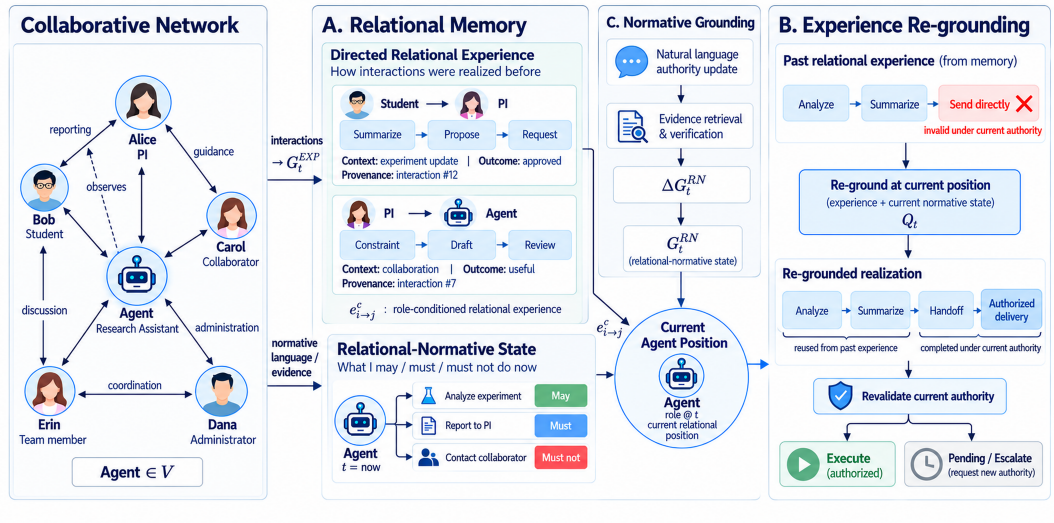

Figure 1: REGROUND framework. The agent is a role-bearing node that learns directed experience from observable human–human and human–agent interactions. A separates this experience from current permissions, duties, and prohibitions. C grounds authenticated natural-language changes through evidence retrieval and verification. B re-grounds experience at the agent's current position, retaining useful structure and searching for an authorized completion. Historical success does not create authority; unresolved goals remain pending.

A: dual relational memory. An experience records an observed relationship:

$$e^c_{i\to j} = (i, j, r_i, r_j, c, \sigma, \rho, \mathcal{P}). \tag{1}$$

Here i, j identify participants and direction, r_i, r_j their roles, and c the context; σ holds a workflow or lesson, ρ an observed outcome or preference, and $\mathcal{P}$ supporting evidence. A language model extracts records from visible dialogue. Admission checks evidence and action identifiers, not generalization correctness. Feedback updates experience without changing permissions.

The normative view stores permissions, prohibitions, duties, role relations, delegation, and support dependencies. Its updates preserve experience. A typed unified store can enforce the same separation as two stores. The system keeps unknown authority, explicit denial, and conflicting evidence distinct.

Trusted inputs. The environment supplies authenticated principals, institutional rules, and action semantics; these are not inferred from unrestricted text. Permissions can depend on actor, operation, object, purpose, time, and supporting relations. Execution creates local artifacts and simulated effects, not external messages. Requesting approval does not grant it.

4 RE-GROUNDING EXPERIENCE

4.1 B: DERIVE A REASONED GAP

The planner replays π^h under current conditions. It checks each step, keeps effects that remain achievable, and traces blocked effects through later dependencies. The result is

$$Q_t = (U_t, B_t, C_t), \qquad L_t \subseteq U_t \times B_t. \tag{2}$$

U_t contains requirements left unmet by the replay. B_t records the authority, state, role, or provenance conditions that block them. C_t contains the constraints on any completion, and L_t links missing requirements to their causes. A revoked step can therefore affect several later steps, not just its own action.

In Figure 2, delivery remains required while direct send is revoked. This distinction directs retrieval toward the missing effect, not the blocked action. The valid Analyze–Summarize prefix can be

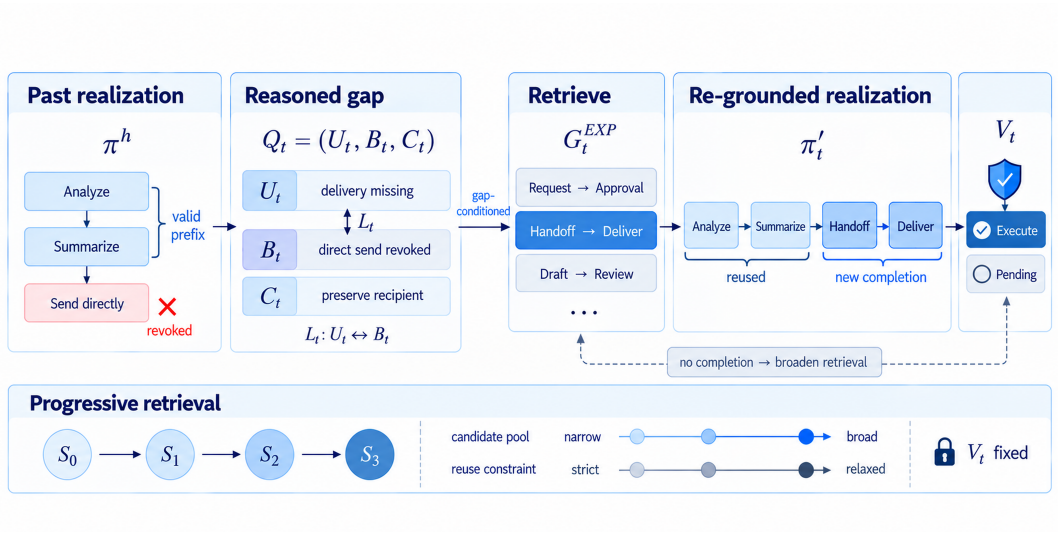

Figure 2: Authority-conditioned experience re-grounding. Replay of π^h yields $Q_t = (U_t, B_t, C_t)$: unmet requirements, their blockers, and completion constraints, with links L_t. The gap guides retrieval from experience G_t^{EXP} (denoted G_t^{E} in the text). Valid structure is selectively reused while search seeks a legal completion. Unsuccessful search broadens candidates and relaxes retained order, never authority checks. The stages are schematic, not the measured five-stage schedule. Validator V_t is fixed within a planning call; π'_t executes only after rechecking current authority. Otherwise the goal remains pending.

reused with a separately authorized Handoff–Deliver fragment, preserving recipient and evidence-source restrictions. This is an illustrative route, not a measured result.

Experience relevance, role compatibility, and executability are distinct. The runtime filters directed records by current source and recipient roles, using explicit role-equivalence mappings when provided. The planner grounds their actions and checks current constraints. Matching responsibilities never creates authority; reusable structure need not be a single prefix.

4.2 Retrieve, Admit, and Complete

The gap changes both the retrieval query and which actions enter search. Requirements and linked blockers rank experiences that could address what is missing. Links select relevant blockers; the ranker aggregates their types and missing facts rather than scoring requirement–blocker pairs separately. In the reference implementation, action/effect matches and lexical overlap map textual strategy steps to public actions. Constraints exclude permanently inadmissible actions. Missing state prerequisites remain candidates for repair rather than being treated as prohibitions.

Search starts with retrieved fragments and their prerequisite closure. Later stages retrieve more experience and release more of the old ordering. The last stage uses a backward causal slice of the full action catalog for the original goal. If S_k is a stage's action set, the cumulative pool is

$$S_k^{\mathrm{cum}} = \bigcup_{j \le k} S_j, \qquad S_0^{\mathrm{cum}} \subseteq S_1^{\mathrm{cum}} \subseteq \cdots \subseteq \mathcal{A}, \quad R_0 \supseteq R_1 \supseteq \cdots, \tag{3}$$

where R_k is the retained historical order. Individual stage sets need not be nested when search switches to the causal fallback.

Each stage restarts a shared $h_{\max}$ heuristic search under the remaining transition-attempt budget. This makes experience a prior: a useful small slice may produce an early solution, but unsuccessful stages spend budget needed by fallback. Expanding candidates never relaxes the validator or changes the original goal. Budget exhaustion is not a proof of impossibility.

A common handler checks relaxed reachability with and without permanent normative restrictions. When a normative cut shows that an external change is necessary, it searches for a legal wait or hold

action. The original requirements remain pending. All methods receive this handler and its budget reservation; its work is included in the reported cost.

4.3 C: GROUND CHANGES IN AUTHORITY EVIDENCE

B requires a current normative state. C updates that state from visible utterances. A language model first proposes an event's force, kind, target, scope, validity interval, and dependencies. The issuer and event time come from the trusted turn record. C retrieves relevant support $H(z, G_t^{\mathrm{RN}})$ and checks whether the issuer can produce the proposed effect:

$$u \xrightarrow{\mathrm{LLM}} z \xrightarrow{H} \text{authority evidence} \xrightarrow{\mathrm{verify}} \Delta G_t^{\mathrm{RN}}. \tag{4}$$

A delegation may support a scoped grant; a request or duty does not. If a grant depends on a role that is later revoked, that support becomes invalid. Independent valid supports survive. Expiry uses half-open intervals. Unsupported or malformed events produce no grant and remain in the trace.

Verification is deterministic within the implemented schema, but an extracted interpretation can still be wrong. Evidence retrieval returns a relevant support closure, not a proven minimum-cardinality set. Main-track extraction reads all visible target turns, so compact verification does not imply lower total history cost. We test evidence locality separately using a pre-folded state.

Execution and persistence. Before planning, the runtime binds protected operations to C's verified state and filters learned experience by role position. It checks each transition again before local execution. New evidence disables action until grounding completes. A verified revocation is saved before experience extraction, so an extraction failure cannot silently restore the old grant. Appendix A records the versioning and failure-path checks.

5 EXPERIMENTAL DESIGN

Data. The supplied benchmark contains AI-generated worlds, frozen into public inputs and private evaluation labels. The A/C track has 37 correlated checkpoints from six worlds, with 171 turn-level events and 106 permission queries. The B track has 432 compiled cases from six stories and a shared planning template. Catalog sizes are 20, 100, and 500 actions; each case supplies 80 structured experiences. Exactly 216 cases permit completion under current conditions, while 216 require an external change.

The supplied B memory is not learned by A. We test learned memory separately and connect all components in a measured integration run. A further multigoal diagnostic uses 40 independently seeded worlds with three perturbations each. Its planner is frozen before execution, although the diagnostic was designed after inspecting the source benchmark.

Comparators. All B methods share the action catalog, authority, validator, experience, causal slicing, and heuristic. Restart searches without preserving the old plan. Strong Repair filters static guards and attempts to retain historical fragments before releasing them. GoalOnly retrieves using the original goals; U uses unmet requirements; UB adds linked blockers; UBC also conditions admission on constraints. The main budget is 10,000 attempted transitions per case.

C compares Full-context, lexical Vector-RAG, Structured-RAG, and extraction plus verification. They use the same language model; the vector baseline uses TF-IDF character n-grams. A compares direct Full-history inference, Unified-Strong, Separated, and Clear-on-change. Unified-Strong and Separated share typed semantics and a decision path. Clear-on-change removes earlier experience when normative state changes.

Models and scoring. The API returns the model identifier `deepseek-flash`; actual requests and usage are recorded. Local Qwen3-4B (Yang et al., 2025) receives the matched API-v2 messages, with 8,192 input tokens and 4,096 output tokens allowed. Failed calls are not retried. Independent evaluators replay plans and distinguish legal completion from pending goal requirements. We report transition attempts, validation work, authority queries, model tokens, and latency separately.

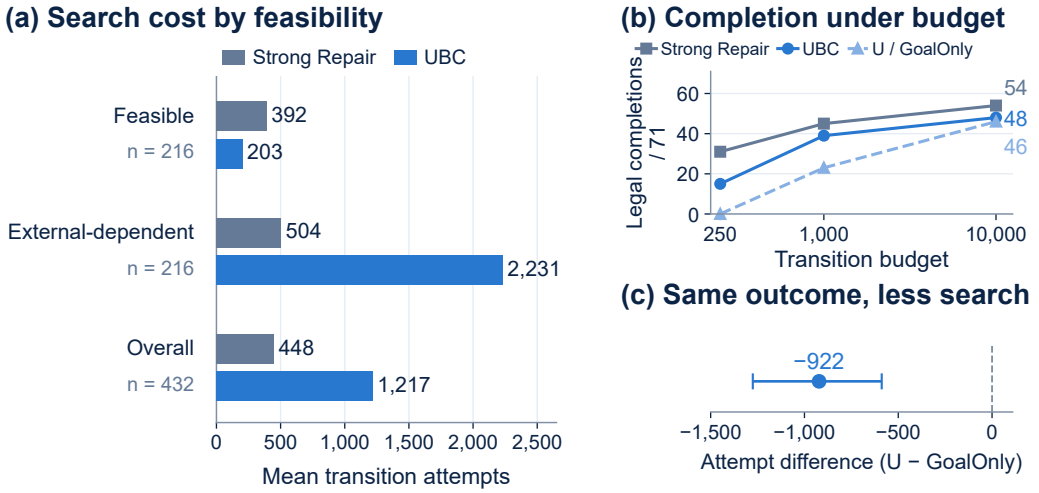

Figure 3: Main empirical results. (a) Search costs on 216 feasible and 216 external-dependent source cases, plus their overall mean. (b) Legal completions among 71 feasible multigoal cases under shared budgets (logarithmic axis); Table 1 retains all methods. (c) At budget 10,000, U minus GoalOnly is -922.03 attempts, with a paired 95% world-bootstrap interval $[-1275.31, -588.09]$. Outcomes agree across all 120 cases and 40 worlds. Search savings do not imply more completions.

Protocols precede each measured run. B v2 and C v2 are engineering revisions evaluated on the same inspected main data, not fresh held-out confirmations. We retain the original runs and describe the changes in Appendix A. Compiled cases and repeated checkpoints are correlated, so their counts are descriptive. Multigoal cost intervals resample the 40 worlds rather than individual perturbations.

6 Results

6.1 B saves work when its first retrieved slice is useful

All six methods complete the same 216 feasible source cases. They also return legal pending dispositions for the 216 cases requiring external change. Independent replay detects no unauthorized transition in a returned plan. The methods therefore differ in search work, not source-benchmark completion.

On feasible cases, UBC uses 202.75 attempts on average, versus 391.50 for Strong Repair, a descriptive 48.2% reduction (Figure 3(a)). Every feasible UBC run finishes in its first retrieved slice. Strong Repair instead searches its global repair set. The records are consistent with a useful early candidate set reducing search, although ranking, admission, and historical ordering are not isolated individually.

The same schedule incurs extra cost when current authority prevents completion. All 216 such UBC runs traverse five completion stages before the common pending handler; Strong Repair uses two. UBC spends 1,747.16 attempts on average in its first three retrieval stages. Total attempts rise to 2,231.16, compared with 504.00 for Strong Repair. Appendix B shows where this work accumulates. Filtering blocked actions prevents their execution but cannot recover budget spent on earlier unsuccessful searches.

Across all 432 cases, UBC uses 1,216.95 attempts against Strong Repair's 447.75, or 2.72 times as many. Mean runtime is also higher, at 36.25 versus 31.70 ms, despite fewer authority queries (200.68 versus 218.44). Search work, query count, and wall time therefore give different cost comparisons.

GoalOnly and U coincide in the supplied single-unmet-goal construction. We test their difference next. Appendix B retains the original B run and its retrieval results.

Table 1: Legal completions among 71 feasible multigoal cases. All 2,160 outputs respect the validator; an empty failure plan does not count as completion.

Method	250	1,000	10,000
Restart	16	41	55
Strong Repair	31	45	54
GoalOnly	0	23	46
U	0	23	46
UB	0	23	46
UBC	15	39	48

Table 2: C on 37 main checkpoints. Event counts require exact force, admission, and activation triples; they do not measure full semantic recovery. V2 is a same-data engineering revision.

Version	Method	Queries /106	Events /171	Exact C /37	Overgrant queries
v1	Full-context	90	150	13	3
v1	Vector-RAG	88	154	12	5
v1	Structured-RAG	88	154	12	5
v1	Ours	100	150	23	3
v2	Full-context	90	150	14	3
v2	Ours	103	160	26	3

6.2 Unmet requirements change search without changing the solved set

We evaluate the frozen planner on 120 multigoal cases from 40 independently seeded causal DAG worlds. Each world has three perturbations and fixed historical experience. Independent breadth-first search finds 71 feasible cases. Every method receives the same public inputs, validator, and transition budgets.

The original goals contain 3.125 facts on average, whereas U contains 1.225. GoalOnly and U rank memories differently on 105/120 cases and choose different first memories on 84/120. At budget 10,000, their outcome statuses agree on every case. Yet U saves 922.03 attempts on average, with a paired world-bootstrap 95% interval of $[-1275.31, -588.09]$. Most aggregate savings occur among the 46 tasks both methods solve. Figure 3(c) shows that the gap changes real search work rather than only its explanation, but does not expand the solved set here.

Adding blocker conditioning (UB versus U) leaves completion counts unchanged at all three budgets; at 10,000, it adds 27.30 attempts on average (95% world-bootstrap interval $[7.58, 51.91]$).

Constraint-aware admission increases legal completions over UB at all three tested budgets. Complete UBC still solves fewer cases than Strong Repair at every budget (Table 1; Figure 3(b)). At 10,000, UBC uses slightly fewer mean attempts (6,256.31 versus 6,344.94), but its difference interval includes zero and completion falls from 54/71 to 48/71.

A saved failure shows how repeated restricted searches can leave little budget for fallback. UBC spends 666 attempts before its final stage and exhausts the remaining 334. Strong Repair releases old structure earlier and completes. Action ordering also differs, so the trace does not isolate budget allocation as the sole cause. Appendix D reports the trace, all intervals, and longer returned plans under U. Lower search effort need not mean lower execution cost.

6.3 C improves state prediction in the tested API configuration at additional cost

In the matched API-v2 comparison, C answers 103/106 permission queries correctly, versus 90/106 for Full-context. Exact C states increase from 14/37 to 26/37 (Table 2). The 13 corrected queries comprise nine unknown states previously classified as denials and four allowed states classified as unknown. Three overgrant queries remain. Thus the gain concerns state distinctions as well as positive permissions.

Backbone	Method	Queries	Events	Exact C	Overgrant	Tokens	Time (s)
API v2	Full-context	90/106	150/171	14/37	3	64,274	71.1
API v2	Ours	103/106	160/171	26/37	3	70,899	79.5
Qwen3-4B	Full-context	80/106	63/171	6/37	22	68,685	1,006.7
Qwen3-4B	Ours	52/106	31/171	6/37	0	67,015	784.1

Table 3: Matched-prompt C results. Overgrant counts queries. Tokens include input and output; time sums client latency and excludes loading. Every planned checkpoint is scored. The 22 Qwen overgrant queries affect 20 items.

Table 4: A on 37 checkpoints. Unified-Strong and Separated share the same typed semantics and decision path. Clear-on-change deletes prior experience; Full-history uses direct model inference.

Method	Behavior /37	Preference /37	Applicability /37	Queries /106
Full-history	28	25	33	83
Unified-Strong	33	26	31	100
Separated	33	26	31	100
Clear-on-change	32	9	11	100

This improvement costs more: 70,899 total tokens versus 64,274, and 79.51 versus 71.05 accumulated client seconds. Main-track extraction still reads visible history before verification. The separate evidence-locality diagnostic isolates structured filtering using pre-folded states. At size 512, Full-context uses 127,630 mean input tokens, ours 816, and Structured-RAG 777. Compression comes from the shared evidence filter, not uniquely from verification (Appendix C.1).

An audit also finds missing issuer/dependency metadata in six initial supports, repeated across 25 checkpoints. Some reference states change when these hidden fields change while public inputs remain fixed. This limits what the public benchmark identifies. Fixed-candidate replays and a later witness correction expose the effect separately; neither replaces the main scores (Appendices C.7–C.8).

6.4 Grounding gains do not transfer to the tested local model

We repeat the 37-checkpoint C comparison with Qwen3-4B. All 62 actual request pairs match API v2 in messages and their 4,096-token output cap. Six empty-history checkpoints per method require no model call. Qwen uses greedy decoding with thinking disabled; provider output constraints and hardware still differ.

With Qwen, ours answers only 52/106 permission queries correctly, versus 80/106 for Full-context (Table 3). Every response parses as JSON, but 19 extraction outputs violate the event contract. Eighteen use invalid event enums and one has incorrect coverage or order. Rejection produces UNKNOWN and many missed-ALLOW errors, including 51 reference-ALLOW queries classified as non-ALLOW. Zero overgrant therefore reflects conservative failure as well as protection against unsupported actions.

A separately frozen thinking configuration does not recover useful grounding. Its three-hour partial run and subsequent first-attempt coverage completion retain all 15 original failures (Appendix E). These outcomes locate a practical bottleneck: verification can check structured candidates only if extraction expresses the relevant event correctly.

6.5 A retains experience; separate physical stores add no measured gain

Unified-Strong and Separated produce the same measured results (Table 4). They share experience extraction and decision rules, so physical separation adds no accuracy gain in this comparison. Clearing experience reduces preference correctness from 26/37 to 9/37 and applicability correctness from 31/37 to 11/37. Permission-query accuracy stays at 100/106 because normative grounding is shared. The useful distinction is which information a change invalidates, not which database stores it. The comparison uses 92 additional A calls; standalone cost must also include cached C grounding.

6.6 Role-conditioned prompting still selects the wrong experience

In the 48-case social diagnostic, 12 cases permit transfer and 36 do not. Role-conditioned memory makes nine appropriate reuses and 15 inappropriate ones; role-agnostic memory makes nine and 23. Conditioning reduces some wrong transfers without increasing correct reuse. It yields 30/48 preferred choices, while always choosing the valid default workflow yields 36/48. Nine useful reuses do not offset 15 harmful selections.

The operational outcome shows the same limitation. Role-conditioned memory completes 11/24 authorized deliveries, compared with 24/24 for the default workflow. All methods preserve the explicitly supplied authority boundary. The failure is therefore selecting position-compatible experience, rather than losing the permission flag. Appendix C.3 reports every method and the diagnostic's construction.

6.7 The complete chain exposes upstream underreach

We evaluate the current system on a 37-checkpoint composite-route simulation using enriched evidence and the revised witness verifier. Each candidate route is one joint-workflow action, unlike the primitive steps in the B benchmark. The system completes 28/31 active duties with no executed overreach.

Three duties stay pending despite having valid reference routes. One extraction repeats an event identifier; the runtime rejects it and retains the duty without searching for a completion. In two other cases, required authority remains unresolved. No transition attempts occur in these three cases, so they do not demonstrate a search-budget failure.

All six no-duty cases correctly produce no action. The earlier integration completed 30/31 duties but executed one unauthorized route. Inputs, verifier, and fresh model draws change together between versions. These are joint A/C/B/runtime measurements, not an isolated B comparison (Appendix C.5).

7 Discussion and limitations

The results separate two decisions often combined under reuse: which experience is useful, and which action is currently permitted. A preserves the former while C updates the latter. B links changes in authority to missing effects and uses those links to guide completion. The source study shows both sides of this choice: a useful first slice saves search, whereas repeated unsuccessful slices add work. The multigoal study shows that this tradeoff persists even when legal solutions exist.

A next step is to use the gap to decide whether further search is worthwhile. A sound external-dependence test could trigger a pending disposition earlier; ordinary search failure would still require a different response. This is not an evaluated policy: the current system uses a fixed stage schedule and invokes its common pending handler after unsuccessful completion search.

Language interpretation creates a different bottleneck. A deterministic verifier can reject malformed or unsupported events, but it cannot repair every mistaken event description. Qwen's UNKNOWN outputs and the pending integration cases make that distinction visible. Likewise, role metadata reduces some inappropriate transfers without ensuring that the chosen experience fits the task.

Our evidence comes from finite synthetic worlds with few independent source scenarios. B receives structured experience, and its lexical grounding does not establish general semantic transfer. The multigoal diagnostic was designed after source inspection; same-data engineering revisions remain exploratory. Missing support metadata limits main-track identifiability, and bounded search can miss feasible completions. The experiments do not establish deployment safety or long-term organizational competence.

Conclusion. ReGround retains collaboration experience while reassessing how the current agent can use it. Its reasoned gap makes that reassessment available to retrieval and search. The measured benefits depend on the task and on the quality of interpreted authority. Keeping experience value, permission, search effort, and goal completion distinct makes those benefits and failures easier to diagnose.

REPRODUCIBILITY STATEMENT

The release includes frozen public/private data, source and protocol hashes, predictions, model-call receipts, independent evaluators, regression tests, and plotting scripts. Original runs remain available alongside versioned revisions. Appendix A records implementation, data details, and reproduction commands.

AI USAGE STATEMENT

AI assistance was used for conceptual development, experiment design, implementation, synthetic diagnostics, experiments, result interpretation, literature search, figure creation, manuscript drafting, and editing. The supplied main dataset was also AI-generated. Numerical claims are checked against saved outputs and independent scoring. Author review and submission decisions remain necessary.

REFERENCES

Blai Bonet and Hector Geffner. Planning as Heuristic Search. *Artificial Intelligence*, 129:5–33, 2001. URL https://www.cs.toronto.edu/~sheila/2542/f10/A1/bonetgeffner-heusearch-aij01.pdf.

Tommaso Cerruti, Mika Okamoto, and Ansel Kaplan Erol. Agent Memory Is a Surface for Endogenous Authorization Laundering. *arXiv preprint arXiv:2609.01836*, 2026. URL https://arxiv.org/abs/2609.01836.

Evan Chen, Shiqiang Wang, and Christopher G. Brinton. Fresh Memory, Stale Plans: Dependency-Scoped Validation for Distributed LLM-Agent Memory. *arXiv preprint arXiv:2609.03340*, 2026. URL https://arxiv.org/abs/2609.03340.

Sizhe Chen, Julien Piet, Chawin Sitawarin, and David Wagner. StruQ: Defending Against Prompt Injection with Structured Queries. *arXiv preprint arXiv:2402.06363*, 2024. doi: 10.48550/arXiv.2402.06363. URL https://arxiv.org/abs/2402.06363.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory. *arXiv preprint arXiv:2504.19413*, 2025. doi: 10.48550/arXiv.2504.19413. URL https://arxiv.org/abs/2504.19413.

Johan de Kleer. An Assumption-Based TMS. *Artificial Intelligence*, 28(2):127–162, 1986. doi: 10.1016/0004-3702(86)90080-9. URL https://www.dekleer.org/Publications/An%20Assumption-Based%20TMS.pdf.

Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents. *arXiv preprint arXiv:2406.13352*, 2024. doi: 10.48550/arXiv.2406.13352. URL https://arxiv.org/abs/2406.13352.

Jon Doyle. A Truth Maintenance System. *Artificial Intelligence*, 12:231–272, 1979. URL https://groups.csail.mit.edu/medg/ftp/doyle/.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. WorkArena: How Capable Are Web Agents at Solving Common Knowledge Work Tasks? *arXiv preprint arXiv:2403.07718*, 2024. doi: 10.48550/arXiv.2403.07718. URL https://arxiv.org/abs/2403.07718.

Maria Fox, Alfonso Gerevini, Derek Long, and Ivan Serina. Plan Stability: Replanning Versus Plan Repair. In *Proceedings of the International Conference on Automated Planning and Scheduling*. AAAI Press, 2006. URL https://cdn.aaai.org/ICAPS/2006/ICAPS06-022.pdf.

Vincent C. Hu, David Ferraiolo, Rick Kuhn, Adam Schnitzer, Kenneth Sandlin, Robert Miller, and Karen Scarfone. Guide to Attribute Based Access Control (ABAC) Definition and Considerations. Technical Report SP 800-162, National Institute of Standards and Technology, 2014. URL https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-162.pdf. Includes updates as of August 2, 2019.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating Multimodal Agents on Realistic Visual Web Tasks. *arXiv preprint arXiv:2401.13649*, 2024. doi: 10.48550/arXiv.2401.13649. URL https://arxiv.org/abs/2401.13649.

Jialong Li and Jialing Zhu. Auditing Self-Evolution in Financial Agents: Capability Gains, Security Drift, and Execution-Interface Mismatch. *arXiv preprint arXiv:2608.17684*, 2026. doi: 10.48550/arXiv.2608.17684. URL https://arxiv.org/abs/2608.17684.

Ninghui Li, John C. Mitchell, and William H. Winsborough. Design of a Role-Based Trust-Management Framework. In *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 114–130. IEEE Computer Society, 2002. URL https://www.cs.purdue.edu/homes/ninghui/abstracts/rt_oakland02.html.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating Very Long-Term Conversational Memory of LLM Agents. *arXiv preprint arXiv:2402.17753*, 2024. doi: 10.48550/arXiv.2402.17753. URL https://arxiv.org/abs/2402.17753.

Bernhard Nebel and Jana Koehler. Plan Reuse versus Plan Generation: A Theoretical and Empirical Analysis. *Artificial Intelligence*, 76(1–2):427–454, 1995. URL https://www.dfki.de/web/forschung/projekte-publikationen/publikation/10370.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. 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*, 2026. doi: 10.48550/arXiv.2509.25140. URL https://arxiv.org/abs/2509.25140. Accepted to ICLR 2026; arXiv version first posted in 2025.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as Operating Systems. *arXiv preprint arXiv:2310.08560*, 2023. doi: 10.48550/arXiv.2310.08560. URL https://arxiv.org/abs/2310.08560.

Zhe Ren, Yibo Yang, Yimeng Chen, Zijun Zhao, Benshuo Fu, Zhihao Shu, Bingjie Zhang, Yangyang Xu, Dandan Guo, and Shuicheng Yan. GateMem: Benchmarking Memory Governance in Multi-Principal Shared-Memory Agents. *arXiv preprint arXiv:2606.18829*, 2026. URL https://arxiv.org/abs/2606.18829.

Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-Based Access Control Models. *Computer*, 29(2):38–47, 1996. URL https://csrc.nist.gov/csrc/media/projects/role-based-access-control/documents/sandhu96.pdf.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language Agents with Verbal Reinforcement Learning. *arXiv preprint arXiv:2303.11366*, 2023. doi: 10.48550/arXiv.2303.11366. URL https://arxiv.org/abs/2303.11366.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent Workflow Memory. *arXiv preprint arXiv:2409.07429*, 2024. doi: 10.48550/arXiv.2409.07429. URL https://arxiv.org/abs/2409.07429.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking Chat Assistants on Long-Term Interactive Memory. *arXiv preprint arXiv:2410.10813*, 2024. doi: 10.48550/arXiv.2410.10813. URL https://arxiv.org/abs/2410.10813.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments. *arXiv preprint arXiv:2404.07972*, 2024. doi: 10.48550/arXiv.2404.07972. URL https://arxiv.org/abs/2404.07972.

Jinghan Xu, Yiyong Xiao, Wanru Shao, Hankai Liu, and Xinjin Li. Memory Provenance Laundering in LLM Agents: A Non-Amplification Firewall for Persistent Memory. *arXiv preprint arXiv:2607.29167*, 2026. URL https://arxiv.org/abs/2607.29167.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic Memory for LLM Agents. *arXiv preprint arXiv:2502.12110*, 2025. doi: 10.48550/arXiv.2502.12110. URL https://arxiv.org/abs/2502.12110.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025. doi: 10.48550/arXiv.2505.09388. URL https://arxiv.org/abs/2505.09388.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. *arXiv preprint arXiv:2210.03629*, 2022. doi: 10.48550/arXiv.2210.03629. URL https://arxiv.org/abs/2210.03629.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents. *arXiv preprint arXiv:2403.02691*, 2024. doi: 10.48550/arXiv.2403.02691. URL https://arxiv.org/abs/2403.02691.

Qiuyang Zhan, Rui Zhang, Sheng Guo, Lepeng Zhao, and Zhuotao Liu. When Memory Becomes Authority: Benchmarking Authority Collapse at the Memory Consolidation Boundary. *arXiv preprint arXiv:2608.01679*, 2026. URL https://arxiv.org/abs/2608.01679.

Guibin Zhang, Muxin Fu, Guancheng Wan, Miao Yu, Kun Wang, and Shuicheng Yan. G-Memory: Tracing Hierarchical Memory for Multi-Agent Systems. *arXiv preprint arXiv:2506.07398*, 2025. doi: 10.48550/arXiv.2506.07398. URL https://arxiv.org/abs/2506.07398.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. MemoryBank: Enhancing Large Language Models with Long-Term Memory. *arXiv preprint arXiv:2305.10250*, 2023. doi: 10.48550/arXiv.2305.10250. URL https://arxiv.org/abs/2305.10250.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A Realistic Web Environment for Building Autonomous Agents. *arXiv preprint arXiv:2307.13854*, 2023. doi: 10.48550/arXiv.2307.13854. URL https://arxiv.org/abs/2307.13854.

A Implementation, data and additional results

The reference system is implemented in Python. A extracts evidence-linked directed records and preserves separate normative state. C uses model event extraction followed by deterministic support-sensitive verification. B uses finite-state planning with shared action validation. The persistent loopback service stores session revisions and locally generated artifacts. Its trusted caller supplies authenticated events and action semantics; it is not a deployed organizational authorization service.

A.1 EXPERIMENTAL PROVENANCE

The main A/C snapshot has version label `20260925_0316` and manifest creation time 03:12:02 UTC on September 25, 2026; the final B snapshot `20260925_0323` was created at 03:23:20 UTC. Provisional B version `20260925_0320` was used for geometry only and is excluded from measured evidence. B's final public JSONL SHA256 is `75cb2802d27b2ac4546c87dff7b329f117d9ea97fc08f3af7c36925c53785e0f`. Public actor loaders reject private-label paths, prohibited nested gold fields and future turns. The independent replay implementation and the source validator agree on 89,712 geometry checks; a separate planner-side parity audit checks 267,840 action/state combinations. These are implementation checks, not empirical success trials.

A.2 ALGORITHM OUTLINE

1. Extract candidate normative events from permitted visible utterances; retrieve supporting authority evidence and verify event force, scope, time and dependencies.
2. Persist the new normative state independently of experience learning. Extract or update directed experience with traceable observed evidence.
3. Normalize the planning schema, bind protected permission identifiers to current verified queries, and filter experience by the agent's current source/recipient position.
4. Replay the old realization under current conditions; construct unmet requirements, blockers, constraints and blocker-to-requirement links.
5. Rank experiences using the chosen goal/gap ablation. Admit compatible fragments and their prerequisites; run bounded search with the shared validator.
6. Expand candidates and release retained order when necessary. If a demonstrated normative cut requires external change, use the common legal pending-disposition handler.
7. Revalidate each planned transition, execute local artifacts/synthetic effects, retain original unmet goals, and append feedback without granting authority.

A.3 VERSIONED ENGINEERING CHANGES

B v1 left external-dependent outputs as null plans, which violated the required serialization/disposition contract without executing an unauthorized action. V2 introduces the same pending handler for every method. It also applies cached static constraint filtering to newly admitted UBC primitives, including prerequisite and fallback admissions. The methods retain the same total budget. When authority blockers exist, ten percent of the budget, capped at 1,000 attempts, is reserved for common terminal handling. All handler work is counted.

C v2 explicitly constrains dependency output types, rejects malformed dependency objects, and resolves uniquely identifiable delegation references during revocation while preserving independent support. Its matched API/local prompts use the same current code and a 4,096-token output cap. V1 results remain the first-run evidence. Improvement between versions is not an isolated causal estimate of a single change.

Integration review independently exposed two defects outside the frozen isolated B experiments: normalizing native inputs after authority binding could restore caller-supplied permissions, and directly forwarding learned records bypassed A's directional position filter. The corrected runtime normalizes first, binds every protected permission from C, then selects position-compatible learned records. Additional failure-path tests cover grounding failure, experience-extraction failure, singular/plural role schemas and stale session revisions. No measured isolated benchmark score was retroactively replaced by these integration tests.

A.4 ALL B METHODS AND COSTS

A.5 RUNNING THE REFERENCE IMPLEMENTATION

The repository's README documents environment setup and the persistent local service. The main commands are `python -B -m unittest discover -s tests -v`, followed by

Table 5: B v2 overall means over 432 compiled cases. All methods complete the same 216 feasible goals and correctly retain 216 external-dependent goals as pending. Internal blocked search proposals are not executed violations.

Method	Attempts	Authority queries	Validator calls	Latency (ms)
Restart	2542.58	2326.49	2592.00	42.91
Strong Repair	447.75	218.44	573.50	31.70
GoalOnly	3068.25	1642.07	3117.67	46.36
U	3068.25	1642.07	3117.67	48.42
UB	2856.68	1647.03	2906.09	46.98
UBC	1216.95	200.68	1313.45	36.25

Table 6: Mean attempted transitions by search phase. Retrieval covers the top-1, top-4 and all-retrieved stages; causal search covers global repair and/or causal expansion with retained structure progressively released. Terminal work is additional to the completion-search trace.

Stratum	Method	Retrieval	Causal	Terminal	Total
Feasible	Strong Repair	0.00	391.50	0.00	391.50
	UBC	202.75	0.00	0.00	202.75
External	Strong Repair	0.00	495.00	9.00	504.00
	UBC	1747.16	475.00	9.00	2231.16

the family-specific `scripts/run_b.py`, `scripts/run_c.py`, `scripts/run_a.py`, and `scripts/run_social.py` with explicit frozen input/output paths. Every measured run creates a manifest before inference and records its actual model identifier and physical calls. The standalone main B result uses no LLM calls. Repeated checkpoint and compiled-case correlation must be preserved in any subsequent statistical analysis.

B WHERE THE PLANNER SPENDS ITS BUDGET

This is a descriptive accounting of the frozen B v2 outputs, not a new run or policy intervention. We join each saved stage trace to independently evaluated feasibility. Stage attempts plus terminal attempts reconcile exactly with the recorded total for every Strong Repair and UBC case. The 2,808 stage-accounting rows also match those records; Table 6 summarizes them.

One-stage success and repeated unsuccessful search. All 216 feasible UBC records finish in `retrieval_top_1`; all 216 feasible Strong Repair records finish in `global_repair`. Neither invokes terminal handling. Their mean attempts are 202.75 and 391.50, respectively, giving the source-stratum reduction of 48.2%. In contrast, all 216 external-dependent UBC records traverse five unsuccessful completion stages, compared with two for Strong Repair. Each external case then uses nine terminal transition attempts under either method.

The external penalty accumulates mainly before UBC reaches its broader causal stages. Each stage starts a new search queue; earlier exploration is not reused as a search frontier. Static admission removes permanently blocked actions but cannot recover attempts already spent. The resulting $4.43\times$ external-case attempt count is not a latency ratio. Nor is the table total computation: external-case mean validator calls, including preprocessing and terminal static checks, are 665.67 for Strong Repair and 2396.86 for UBC. A legal pending plan keeps the original goal pending; it does not complete it.

These records support the explanation that a useful small first slice saves work, whereas repeated restricted searches can consume fallback budget. They do not identify separate causal effects of ranking, candidate size, retained order, caching and action ordering. The uniform one-stage source successes also limit generalization: feasibility alone does not guarantee a benefit on the multigoal diagnostic.

Why U can save work without additional completions. At multigoal budget 10,000, U and GoalOnly have identical statuses on all 120 cases. Summing the saved paired attempts gives 110,643

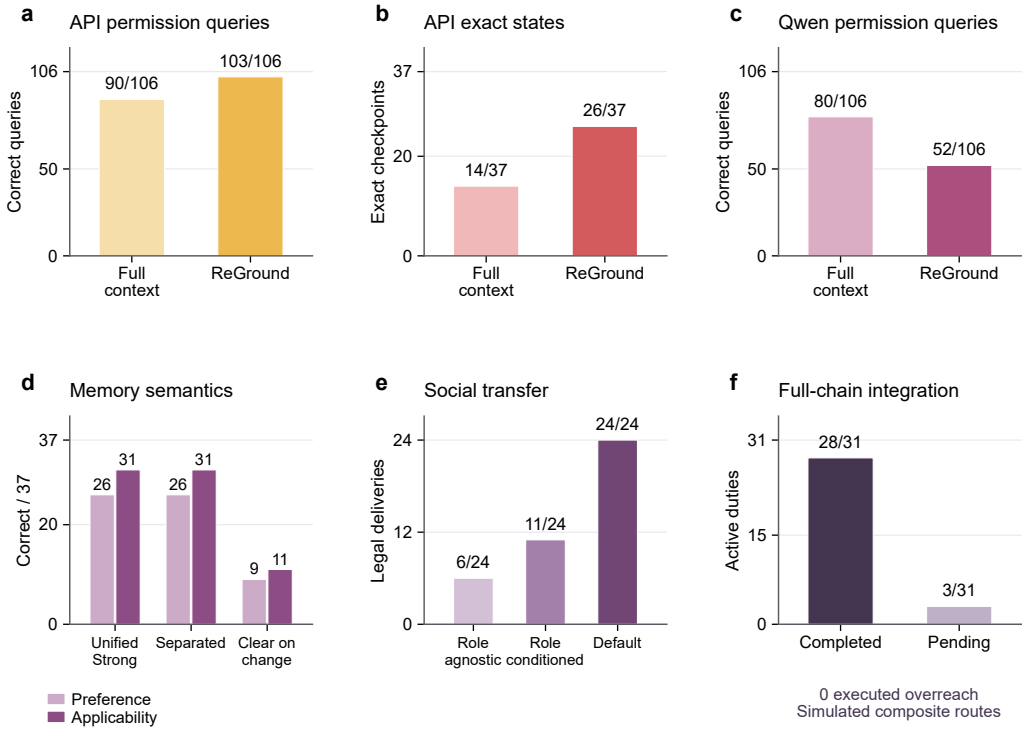

Figure A1: Component and integration diagnostics from existing runs. (a–b) Matched API-v2 permission queries and exact normative states improve with extraction plus verification. (c) The same pipeline lowers greedy-Qwen query correctness. (d) Typed unified and separated memory tie; clearing experience loses preference and applicability judgments. (e) Social delivery remains below the default workflow. (f) The current witness-system integration completes 28 of 31 active duties, retains three as pending, and executes zero overreach. Panels summarize different protocols, not a controlled end-to-end ablation; Tables 2 and 4 retain detailed results.

fewer attempts under U, or 922.025 per case. Of these savings, 88,406 occur among their 46 common successes, 14,623 among 61 common budget failures, and 7,614 among 13 common external-authority outcomes. Most savings therefore reduce work on tasks both methods solve rather than move the completion boundary. This is not a uniform per-case gain: two common successes cost more under U. Appendix D reports the world-level uncertainty and the longer-plan tradeoff.

All quantities in this section come from the saved `results.jsonl`, independent `evaluation.json`, and cost tables under `b_v2_0323` and `b_multigoal_v1`. No actor, outcome, or original score was changed for this accounting.

Original retrieval result. In B v1, top-1 retrieval recall rises from 0.00347 for U to 0.28472 for UB and UBC. The latter two have identical rankings because constraints act after retrieval, when actions enter search. Better retrieval does not establish lower total cost: the original planner also loses to Strong Repair overall. V1 leaves external-dependent cases without the required pending output; the common handler in v2 corrects that output contract. We retain this distinction rather than comparing the two versions as a controlled intervention.

C Component and Integration Diagnostics

Figure A1 summarizes the component and integration results; Figure A2 separately examines context scaling.

C.1 EVIDENCE-LOCALITY DIAGNOSTIC

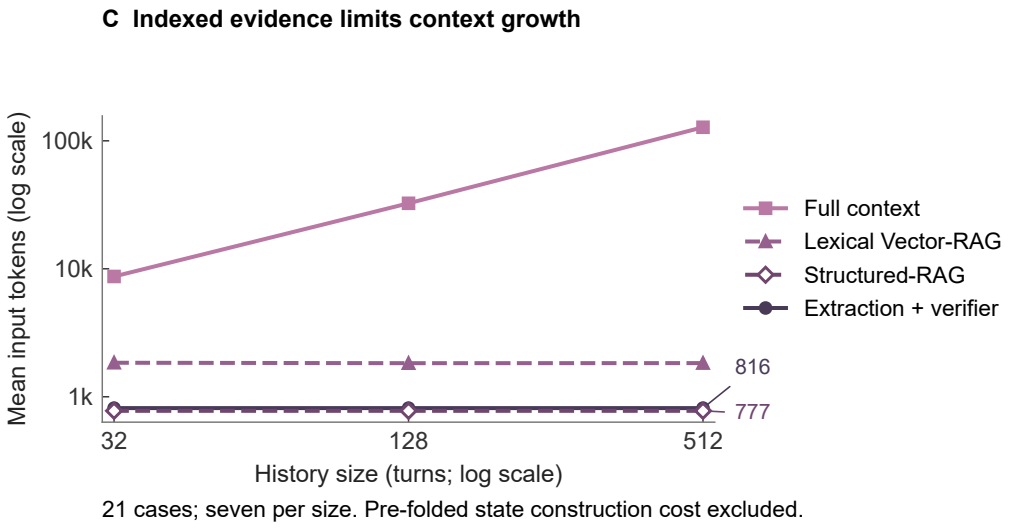

Figure A2: Evidence locality in the synthetic pre-folded-state diagnostic. Structured retrieval and ours share evidence filtering; compact context is not a unique benefit of verification. Both axes are logarithmic. Initial-state construction cost is excluded.

The 21 diagnostic cases consist of seven contexts at sizes 32, 128 and 512. They include same-operation/scope distractors. Full-context, vector, structured and ours yield 11, 13, 19 and 21 correct permission decisions respectively. Their event triple counts are 9, 9, 18 and 18; overgrant counts are 9, 6, 2 and 0. Ours' remaining event errors occur in request-only contexts. This controlled construction does not support a claim of perfect general language understanding.

C.2 BACKBONE AND MEMORY COMPARISONS

Development fixtures. The actual-model integration fixture separately exercises observation, revocation, planning, and local artifact execution. The API fixture passes. With the same fixture, Qwen's two calls bind grants to *PI authorizes Agent* instead of the protected send or handoff operations. The goal remains pending because the inferred state cannot authorize the route. This development failure concerns semantic binding, not benchmark accuracy.

C.3 SOCIAL-TRANSFER DIAGNOSTIC

Table 7: All social diagnostic methods. Reuse is evaluated separately on applicable and inapplicable cases; default workflow is always valid.

Method	Preferred /48	Good reuse /12	Bad reuse /36	Delivery /24
No experience	36	0	0	24
Raw dialogue	27	10	19	11
Role agnostic	22	9	23	6
Role conditioned	30	9	15	11

The synthetic operational rule requires matching sender responsibilities, recipient responsibilities and scope. It is not a universal rule that knowledge from reversed relations is useless. Positive cases permit equivalent responsibilities with different titles. There are 12 positive and 36 negative cases, so unconditional standard fallback obtains 75% preferred-choice accuracy. The two distilled conditions share 46 actual extraction calls and each use 48 selection calls; all four methods together use 238 physical calls. Input/output totals are 193,098/15,160 tokens. Hypothetical clarification-round counts are stipulated by construction, not measured interaction latency.

C.4 COMPLETE-CHAIN ROUTE INTEGRATION

This diagnostic connects fresh A learning and C grounding to state binding, B completion, and local execution on all 37 supplied A/C checkpoints. Each public realization is one composite joint-workflow action. It retains the actual performers and operation–permission bindings, but differs from B's primitive-action benchmark. The diagnostic has one system arm and uses previously inspected data.

Independent evaluation finds 34/37 correct task behaviors and 30/31 legal completions of active duties. All six no-duty cases correctly produce no action. Two behavior errors choose legal alternatives to the preferred route; one executes an unauthorized simulated action and falsely reports completion. C answers 103/106 queries correctly, with three overgrants, and obtains 158/171 event triples and 37/37 duty judgments.

All 37 outputs are present without stage exceptions. The run uses 62 physical API calls, 91,503 input tokens, 33,471 output tokens, and 133.08 accumulated client seconds.

The original goal remains in every active-duty planner input and report. In the unsafe case, however, incorrect inferred authority leads the system to discharge an unfulfilled obligation. Keeping the goal's identity is therefore different from completing it legally. All actions are local simulations, and every active duty has a legal reference route. The result exposes how an upstream grounding error can propagate through the full chain.

C.5 CURRENT WITNESS-SYSTEM ROUTE ACCEPTANCE

The current system completes 28/31 active duties, leaves three pending, and executes no unauthorized simulated action. This run uses enriched initial evidence, the witness verifier, current state binding, B v2, and fresh A/C inference. All 37 checkpoints are retained, including one invalid C interpretation.

There are 62 physical calls, with 92,307 input tokens, 33,933 output tokens, and 133.28 accumulated client seconds. No transport exception occurs.

Table 8: Descriptive complete-chain comparison. Routes remain composite joint actions.

Metric	Original route	Witness route
Exact behavior /37	34	32
Legal duty completion /31	30	28
Executed overreach	1	0
False completion	1	0
Legal completion or pending /31	30	31
Permission queries /106	103	101
Overgrant queries	3	0
Event triples /171	158	152

Two pending outputs request authorization; one defers after an invalid interpretation. No request is treated as a grant. Every duty has a legal reference route, so these three cases are underreach rather than inherently infeasible tasks. All six no-duty cases correctly produce no action, and no external operation occurs.

The versions differ in metadata, verifier/runtime, and model draws. The comparison describes their joint behavior, not the causal effect of one correction. The lower overreach count accompanies fewer completed duties.

C.6 DESCRIPTIVE AUTHORITY–CONVENTION CELLS

The supplied core has 24 checkpoints: six each in Stable, Authority-only, Convention-only, and Joint conditions. Thirteen Common checkpoints are reported separately. Each core condition contains the same six source worlds; these counts do not treat checkpoints as independent trials or estimate a causal factorial interaction. All cells and methods are retained. A uses its original C-v1

Table 9: A by cell. Entries are behavior exact / preference exact / permission queries exact, with denominators shown in the cell heading. Unified and Separated share typed decisions.

Method Items / queries	Common 13 / 38	Stable 6 / 17	Authority 6 / 17	Convention 6 / 17	Joint 6 / 17
Full history	8/7/32	3/4/14	6/2/12	5/6/13	6/6/12
Unified	12/6/37	4/5/13	6/3/17	5/6/16	6/6/17
Separated	12/6/37	4/5/13	6/3/17	5/6/16	6/6/17
Clear on change	12/0/37	4/1/13	6/0/17	4/4/16	6/4/17

Table 10: C permission-query correctness by cell. Common has 38 queries; each core cell has 17. All results use the original public inputs.

Version	Method	Common	Stable	Authority	Convention	Joint
API v1	Full context	34/38	15/17	13/17	15/17	13/17
API v1	Vector-RAG	34/38	15/17	12/17	15/17	12/17
API v1	Structured-RAG	34/38	15/17	12/17	15/17	12/17
API v1	Ours	37/38	13/17	17/17	16/17	17/17
API v2	Full context	34/38	15/17	13/17	15/17	13/17
API v2	Ours	37/38	16/17	17/17	16/17	17/17
Qwen greedy	Full context	33/38	13/17	10/17	15/17	9/17
Qwen greedy	Ours	24/38	4/17	9/17	6/17	9/17

cache. The machine-readable decomposition additionally reports duty, event, exact-state, overgrant and execution-error counts. These descriptive slices do not turn post-review API v2 or the model comparison into held-out confirmation.

C.7 Fixed-candidate metadata replay

Adding missing authority evidence removes some overgrants but also creates new errors under the original verifier. The cause is a scope fallback: when a dependency has no explicit jurisdiction, the verifier uses the operation's purpose. That purpose need not match the scope of the supporting relation.

We examine this behavior without changing the extracted events. Two saved candidate sets, from original and enriched inputs, are replayed under both evidence conditions using the frozen verifier. This gives 148 outputs without model calls. The 74 outputs with their original candidate/evidence pairing reproduce the saved states exactly.

Table 11: Deterministic replay. Each row contains 37 checkpoints and 106 permission queries.

Candidate source	Evidence	Queries	Overgrants	Events /171	Exact C /37
Original API	Original	103/106	3	160	26
Original API	Enriched	94/106	0	160	22
Enriched API	Original	100/106	3	154	26
Enriched API	Enriched	92/106	0	154	22

For original candidates, enriched evidence corrects three overgrants but creates twelve new query errors. For enriched candidates, it makes three corrections and eleven new errors. Event and duty labels remain unchanged within each pair. The scope mismatch removes some valid ALLOW and explicit DENY supports, explaining why more evidence can reduce accuracy here. The crossed replay isolates verifier behavior for fixed candidates. It does not predict what a model would extract under a different prompt.

C.8 VERSIONED DEPENDENCY-WITNESS CORRECTION

We revise the scope rule after inspecting the metadata replay. When dependency scope is implicit, the new verifier resolves it from a unique trusted initial or issuance witness. It preserves explicit scopes and tracks issuer and subject dependencies separately. An explicit tuple can become active later. An unresolved implicit scope cannot be supplied by a later unrelated role.

This is a stricter reference-system rule, not a rule proven by the benchmark labels. The supplied evaluator checks principal and relation validity without jurisdiction. It therefore cannot establish the correctness of the new jurisdiction check.

We replay the same 148 candidate/input pairs with this version, without new model calls or changes to the old verifier. All outputs and candidate identities pass independent checks.

Table 12: Witness-v3 replay on already-inspected data. Each row has 37 checkpoints.

Candidates	Evidence	Queries /106	Overgrants	Events /171	Exact C /37
Original	Original	103	3	160	26
Original	Enriched	106	0	160	27
Enriched	Original	100	3	154	26
Enriched	Enriched	101	0	154	27

On identical enriched evidence, original candidates gain twelve correct queries without new errors. Enriched candidates gain ten but acquire one new ALLOW-to-UNKNOWN error, for a net gain of nine. Two already-invalid states remain in the denominator. Even the 106/106 query result has only 27/37 exact C states because event errors remain. This posthoc correction improves fixed-candidate behavior but is neither regression-free nor held-out evidence.

D FROZEN MULTIGOAL DIAGNOSTIC

Data and execution boundary. The diagnostic has 40 independently seeded causal DAG worlds (seeds 620100–620139), each with three perturbations and unchanged historical experience. Public episodes provide the initial state, original goals, historical plan, primitive catalog, current authority and memories. Private BFS answers and repair/relevance labels are excluded from actor inputs. Independent BFS identifies 71 feasible and 49 infeasible episodes. Before running the actor, geometry checks validate 120 historical realizations, 750 alternative historical trajectories and 5,934 action-guard comparisons.

The planner is frozen before this run: its SHA-256 begins `e9833a3970b7eb00`, and the dataset hash begins `2ad5b1a4ea6141a2`. The manifest retains full hashes, immutable code and the separately frozen adapter. Six methods run on all 120 cases at budgets 250, 1,000 and 10,000: 2,160 calls, with deterministic rotation of method order and increasing budgets. The run takes 75.83 seconds without exceptions, below its predeclared 30-minute cap. Independent replay follows actor completion. No tuning or rerun follows the result review.

Figure 3(b) summarizes completion for Strong Repair, UBC, and U/GoalOnly. Table 1 retains all six methods; UB shares the same completion counts as U and GoalOnly.

Accounting and uncertainty. The budget counts primitive transition attempts, including rejected proposals. Total validator calls and authority queries also count impact analysis, static screening, retrieval admission and final replay. On authority-blocked cases, every method reserves at most 10% of the transition budget (capped at 1,000 attempts) for the same pending handler. Requests do not grant authority; pending status does not satisfy the original goal.

All-case means average perturbations within each world, then the 40 worlds. Paired differences compare methods at the same world and budget. Descriptive 95% intervals use 10,000 world-bootstrap resamples, seed 240925. Feasible-only means average available feasible perturbations within the 37 eligible worlds. Related perturbations are not independent samples. Costs include failures and must be read alongside completion: a cheap failure is not an efficient solution. These intervals do not establish real-world generalization or a multiple-comparison discovery claim.

Table 13: Paired all-case differences in primitive attempts (left method minus right). Negative values mean fewer attempts, not necessarily better completion. Intervals resample the 40 worlds.

Comparison	Budget	Mean difference	95% interval
U − GoalOnly	250	0.00	$[0.00, 0.00]$
	1,000	-0.58	$[-1.51, 0.00]$
	10,000	-922.03	$[-1275.31, -588.09]$
$UB - U$	250	0.00	$[0.00, 0.00]$
	1,000	-0.29	$[-1.03, 0.27]$
	10,000	27.30	$[7.58, 51.91]$
$UBC - UB$	250	-3.27	$[-5.21, -1.57]$
	1,000	-49.36	$[-65.93, -33.47]$
	10,000	-529.89	$[-818.83, -279.67]$
UBC − Strong Repair	250	11.69	$[7.61, 16.02]$
	1,000	101.71	$[72.47, 131.27]$
	10,000	-88.63	$[-421.25, 253.14]$

Retrieval and validation cost. GoalOnly and U differ in full and top-five rankings on 105 cases, and in top-one rankings on 84. Across 120 cases, the top-three count of memories concerning only already-completed goals falls from 198 to 93. This public-only redundancy measure is not Recall@k; there are no private relevance labels. At budget 10,000, U minus GoalOnly is -920.75 total validations (95% interval $[-1273.49, -587.17]$) and -639.33 authority queries ($[-852.03, -435.78]$). Status agrees on every case, while mean plan length increases by 1.275 actions. Some earlier successes retain more historical steps: search savings need not reduce execution cost.

At budget 250, UBC uses fewer transition attempts than UB but more total validations (285.53 versus 262.28): static filtering has a cost. At 1,000 and 10,000, total-validation differences favor UBC over UB by 24.24 and 502.42 checks, respectively. Against Strong Repair at 10,000, the total-validation difference is -86.52 with interval $[-419.22, 255.38]$, while completion is lower. None of these comparisons supports general planner dominance.

Illustrative traces and failure mechanisms. These cases were selected after the run using recorded rules; they illustrate behavior, not new experiments or representative samples. The largest U saving among common successes is `I038eb06a452292`: U solves its first 13-action slice in 754 attempts, while GoalOnly starts with a different 14-action slice and needs 7,341 attempts across successive stages. The changed query therefore changes admitted actions and actual search work.

For `I09c5e8f613201e` at budget 1,000 (the first lexicographic UBC failure with Strong Repair success), UBC spends 166, 166, 167 and 167 attempts in four restricted stages, then exhausts its remaining 334 attempts. Strong Repair spends 333 attempts retaining old structure and completes after another 567 with that structure released. Their final searches share the same 27-action set but use different action orders, as reconstructed from the saved rankings and frozen sorting rule. The trace shows delayed fallback and a smaller remaining budget; it does not isolate budget allocation as the sole cause of failure.

For `I017594665577cb` at budget 1,000 (the first lexicographic UBC success with UB failure), both rank the same first three memories. Constraint admission removes two blocked primitives from the final slice: 32 rather than 34 actions. UBC solves in 728 attempts and 780 total validations; UB exhausts 900 completion attempts. This shows where constraints affect search, without claiming that static filtering itself is new or that UBC beats the stronger filtered comparator.

Pending states and claim boundary. Every method reports that external authority is needed for 13 independently infeasible cases at each budget. All reported pending-requirement counts agree with independent replay. At budget 10,000, UBC also returns without a completion on 36 infeasible and 23 feasible cases. Without reference pending routes, we claim neither pending-route correctness nor institutional resolution. Finite-model infeasibility need not persist after authority changes.

This synthetic diagnostic was designed after inspecting the source data, although its actor and protocol were frozen before its outcomes. It does not replace the source v1/v2 runs or provide external held-out confirmation. Heuristic search, plan repair, causal slicing and static filtering remain standard mechanisms. The results identify conditional effects of the gap interface and limitations of its current composition.

E BOUNDED THINKING CONFIGURATION AND COVERAGE COMPLETION

Thinking does not resolve the extraction failures in this bounded diagnostic. The combined full-context and extraction conditions answer 84/106 and 37/106 queries correctly. Zero overgrant in the latter accompanies low query accuracy. We retain the original time-limited result separately from coverage completion below.

After the greedy Qwen failure, we froze a separate configuration using the official Qwen3-4B thinking-mode sampler: temperature 0.6, top-p 0.95, top-k 20 and min-p 0. Each item/method receives a fixed SHA256-derived seed from base seed 20260925. Message arrays and the C source snapshot match API v2; input and total generated output are capped at 8,192 tokens, with reasoning charged to the latter. Each call receives at most 300 seconds. Only text after closing token 151668 is parsed. Mode, sampling and output cap change together, so this is not an isolated thinking effect.

The primary run stopped at its three-hour deadline (10,800.187 seconds including cleanup), with 72/74 outcome rows, 60 forwards, 45 parsed responses and 15 generation/parsing failures. It remains **partial**. Every planned 37-checkpoint/106-query denominator is retained. The last started extraction received 212.184 seconds before the global stop and failed; it was not retried.

Method	Queries	Events	Exact C	Overgrant	Parsed/calls	Tokens	Time (s)
Three-hour primary run							
Full context	81/106	45/171	6/37	6	28/30	129,215	3,864.7
REGROUND	37/106	21/171	6/37	0	17/30	182,471	6,927.4
Combined coverage, including the two supplemental calls							
Full context	84/106	49/171	6/37	6	29/31	132,882	3,951.8
REGROUND	37/106	21/171	6/37	0	18/31	188,204	7,121.6

Table 14: Thinking configuration, with separate primary and supplemental views. Tokens include input, reasoning and final output; time sums client latency, including failed calls. Parsed JSON does not imply a valid event contract. The combined view is not a result completed within the three-hour policy.

A coverage-only supplement was frozen before primary effectiveness was read and launched after independent review. It makes only the two never-started first attempts, preserving every original row and failure. The combined view has 74 outcome rows, 62 forwards, 47 parsed responses and the same 15 failures; 11 of the parsed extraction outputs violate the typed contract. Of 235,131 generated tokens, 194,540 are reasoning tokens. Eleven failed generations lack the closing marker; four fail final JSON parsing. All original failures remain unchanged. The combined view does not establish a benefit from this configuration, and it is not a result completed within three hours. Scores, costs, origins, and message/source checks are stored separately for both views.