REVERSIBLE EVIDENCE-GRAPH ACTION PROJECTION FOR FROZEN LANGUAGE AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Grounded language-agent systems must act under delayed and reversible evidence, because observations confirm effects only after execution and later interactions can invalidate established facts. Overcoming this difficulty requires an evidence-synchronized action interface backed by transactionally verified and revocable state, rather than fluent reasoning and syntactic executability alone. Planning and reasoning agents are well suited to adaptive decision generation, yet their reliance on textual histories leaves stale and premature actions available when interaction evidence changes, especially in environments that enumerate grounded actions. Within this setting, StructuThink addresses task ordering through transition knowledge, but its task graph is derived from expert refinements and does not revise the environment-supplied feasible set from delayed receipts and revoked support. We therefore propose Reversible Evidence-Graph Action Projection (REGAP), which maintains receipt-gated typed state and revocable evidence to construct a compact, nonempty action interface for a frozen selector, together with guarantees for interface totality, evidence soundness, and conditional path preservation. Experiments on embodied household and scientific-interaction benchmarks show that REGAP substantially improves completion and context efficiency over complete-action, planning, and reasoning baselines.

1 INTRODUCTION

Language models increasingly operate tools and embodied environments by interleaving decisions with observations (Yao et al., 2023b; Ichter et al., 2023; Huang et al., 2023; Liang et al., 2023). In long-horizon interaction, an executable action can commit an unconfirmed effect, rely on evidence that later becomes invalid, and destroy a prerequisite needed downstream. The central interface mismatch is that environment-supplied actions are executable, while grounded progress requires actions whose prerequisites remain supported by current evidence. Solving it requires an evidence-synchronized action interface backed by transactionally verified and revocable state.

Reflection, search, and executable planning improve proposal generation and repair (Shinn et al., 2023; Yao et al., 2023a; Hao et al., 2023; Sun et al., 2023). These methods are well suited to deliberative decision making, yet their textual state can preserve requested effects before confirmation and retain conclusions after their support disappears. Grammar-constrained decoding and shielding enforce structural validity and safety predicates (Scholak et al., 2021; Poesia et al., 2022; Alshiekh et al., 2018), while static constraints do not track pending effects, revocable evidence, and result-dependent progress. This narrows the problem to structured controllers that update semantic feasibility at every environment step.

KnowAgent constrains planning with an action knowledge base, and StructuThink builds subtask chains from task-transition knowledge (Zhu et al., 2025; Zhao et al., 2025). StructuThink addresses task ordering through expert refinements, but its transition graph does not revise the environment-supplied feasible set from delayed receipts and revoked support. Transactional agent systems provide commit boundaries for effects and memory (Chen et al., 2026c; Li et al., 2026); grounded action selection additionally requires a nonempty decision set synchronized with the current interaction state.

We therefore propose Reversible Evidence-Graph Action Projection (REGAP), an inference-time controller that compiles interaction receipts into typed pending effects and revocable evidence under

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.

a fixed task-family schema. Before each decision, REGAP presents an unchanged model with a compact, nonempty projection of the supplied executable set. We establish interface totality, receipt and evidence soundness, conditional path preservation, and linear action-count complexity at fixed controller-state size.

Across ALFWorld and ScienceWorld under paired frozen-model protocols, REGAP substantially improves completion and context efficiency over complete-action, planning, and reasoning baselines. ScienceWorld mechanism controls identify contributions from compact candidate presentation, receipt-gated updates, revocation, and language-conditioned selection within the projected set. We make three contributions.

1. We introduce a constructive inference-time controller that turns receipt-confirmed, reversible interaction evidence into a nonempty grounded-action projection for a frozen language agent, without model training, an auxiliary model, or an extra model call.
2. We formalize receipt-aware state updates, reversible evidence maintenance, and nonempty action projection, with guarantees for interface totality, evidence soundness, conditional path preservation, and implementation complexity.
3. We provide paired evaluations across household and scientific interaction, including prospective confirmations, planning and reasoning comparisons, cross-model replications, mechanism controls, selector attribution, and exact implementation audits.

2 Related Work

Reasoning and planning agents. ReAct interleaves reasoning with environment actions (Yao et al., 2023b), while Reflexion stores verbal feedback across attempts (Shinn et al., 2023). Tree of Thoughts, RAP, and language-agent tree search allocate more inference to planning (Yao et al., 2023a; Hao et al., 2023; Zhou et al., 2023). AdaPlanner generates executable plans and repairs them using environment feedback (Sun et al., 2023); Code as Policies and ProgPrompt similarly express control as programs (Liang et al., 2023; Singh et al., 2023). REGAP complements this family by changing the feasible set presented to a frozen selector at every step.

Structured decision knowledge. KnowAgent constrains planned action paths with an action knowledge base and then performs knowledgeable self-learning (Zhu et al., 2025). StructuThink learns a task-transition knowledge graph from expert trajectories and uses subtask chains to guide a prompt-reflect-act executor (Zhao et al., 2025). These methods encode task order, whereas REGAP projects the environment-enumerated set at every state and admits persistent effects only after typed receipts. Its contribution is per-state projection from receipt-gated and revocable evidence without synthesizing training trajectories and learning a transition graph.

Grounding and tool use. Affordance scoring connects instructions to robot skills (Ichter et al., 2023), and observation-conditioned monologues feed execution feedback into embodied reasoning (Huang et al., 2023). Toolformer, MRKL, and ToolLLM expose external capabilities to language models (Schick et al., 2023; Karpas et al., 2022; Qin et al., 2023). Voyager builds a reusable skill library (Wang et al., 2023), while HaReCAP compiles recurring grounded decisions into abstainable reflex rules (Liu et al., 2026). REGAP differs by retaining the model as the action selector and projecting the supplied executable set from online evidence.

Constrained decisions and state integrity. PICARD, Synchromesh, LMQL, and guided generation constrain token sequences with parsers, grammars, and declarative predicates (Scholak et al., 2021; Poesia et al., 2022; Beurer-Kellner et al., 2023; Willard & Louf, 2023). Safe reinforcement learning uses shields and verifiable policies to block unsafe transitions (García & Fernández, 2015; Alshiekh et al., 2018; Bastani et al., 2018). AgentSpec applies customizable runtime predicates before execution (Wang et al., 2026). Agent Behavioral Contracts and policy algebra formalize preconditions, path admissibility, governance, and recovery, while AgentFlow and path-based governance mediate labeled flows or proposed actions using execution history (Bhardwaj, 2026; Tripathi et al., 2026; Shivakumar et al., 2026; Kaptein et al., 2026). Sentinel counterfactually simulates graph mutations before an Allow, Block, or Clarify decision, and PolicyGuide compiles policy into a persisted workflow graph with verifier-proposed remediation (Wu & Gong, 2026; Kang et al., 2026).

These systems target compliance or security and mediate a proposal or user turn; REGAP constructs a nonempty task-progress decision set from a supplied candidate set and analyzes solution-path retention without auxiliary inference. Retrieval and persistent memory provide external state to models (Guu et al., 2020; Lewis et al., 2020). MAGE isolates invalid execution branches, PlanFence revalidates plan dependencies after shared-state changes, and Cordon and MemTX add transaction semantics to effects and beliefs (Chen et al., 2026b;a;c; Li et al., 2026). The distinguishing combination is per-state task-progress set projection from receipt-gated and revocable evidence, with a nonempty interface and no auxiliary inference.

3 Problem Setup

Definition 1 (Grounded language-agent process)**.** *A grounded interaction is a goal-conditioned partially observed process (Sutton & Barto, 2018). At decision step t, the environment returns an observation bundle o_t, including any persistent auxiliary fields exposed by the interface, and a nonempty supplied set $\mathcal{A}_t$ of actions accepted as executable by that interface. The adapter derives a typed receipt $r_t = R_\Gamma(S_{t-1}, o_{t-1}, a_{t-1}, o_t)$ from the latest transition feedback and the pre-action controller state, with r_0 null. After the controller incorporates r_t, a frozen language policy π selects a_t from a supplied subset of $\mathcal{A}_t$.*

Definition 2 (Action projection)**.** *Let $\mathcal{U}$ be the action universe and $\mathcal{S}$ the controller-state space. Following the constraint-filtering view used in shielding (Alshiekh et al., 2018), an action projection is a map $P_\Gamma : \mathcal{S} \times 2^{\mathcal{U}} \to 2^{\mathcal{U}}$ satisfying $P_\Gamma(S, A) \subseteq A$. At step t, its output is $\widehat{\mathcal{A}}_t = P_\Gamma(S_t, \mathcal{A}_t)$, and the projection is feasible when this set is nonempty.*

The fixed task schema Γ specifies phase precedence, and the dynamic controller state is $S_t = (B_t, G_t, Q_t, Z_t, M_t)$. Here B_t contains typed runtime bindings, G_t is a reversible evidence graph, Q_t is either one pending typed effect or $\perp$, and Z_t contains phase and result registers; write $D_t = (B_t, Z_t)$ for the non-graph controller state. Before updating memory, the adapter evaluates $\Delta^M_\Gamma(S_t, a_t, o_{t+1}) \in \{0, 1\}$. If it is zero, M_{t+1} records (κ_t, a_t), where κ_t is derived from persistent interface fields and excludes M_t. Its certificate applies only to Γ-configured families at the same key. REGAP-specific certificates are introduced next.

The interaction interface separates two notions that are often conflated. The supplied set $\mathcal{A}_t$ represents executable candidates and, when enumerated directly by the environment, membership only states that an action can be issued. Semantic feasibility asks whether issuing it is supported by the current goal, verified effects, and unresolved prerequisites. The complete set can therefore contain actions that are syntactically valid yet premature, contradictory, repetitive, or dependent on revoked evidence. REGAP never invents an action outside $\mathcal{A}_t$. When an environment does not enumerate actions directly, templates, affordance enumeration, or grammar sampling may construct the candidate set; all containment and path claims then apply to that supplied set rather than to unenumerated actions.

An episode succeeds when the official environment signal is reached within budget. Our controller is evaluated by four properties. Commit fidelity requires every stored world effect to follow a matching receipt. Revocation consistency requires evidence derived from a disconnected object to disappear. Conditional reachability requires the projected set to retain a witness-compatible solution when the schema covers one. Practicality requires projection to use no extra neural call and to scale approximately linearly with the enumerated action set.

4 Method

REGAP operates between the environment and an unchanged action selector. Each interaction runs three stages. *Update* derives a receipt and mutates typed state, *Certify* evaluates six exclusion classes, and *Project* applies the total interface before serializing the retained set into the original action field. Figure 1 summarizes this loop.

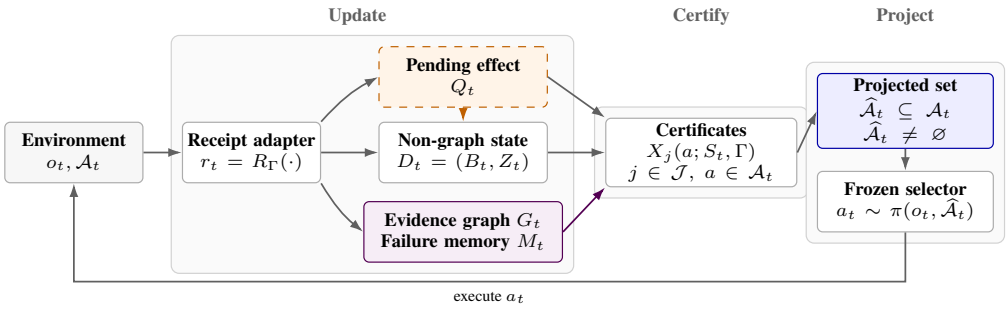

Figure 1: REGAP updates typed state from receipts, certifies exclusions, and projects a nonempty action set for the frozen selector.

4.1 Receipt-Aware State Update

Given pre-action state S, the effect compiler $F_\Gamma(a; S)$ returns a typed partial assignment of world effects to the non-graph state $D = (B, Z)$. Receipt-backed graph updates are defined separately in Section 4.2. After executing a_t, let $u_{t+1} = \tau(r_{t+1}) \in \{\text{resolved}, \text{clarify}, \text{success}, \text{other}\}$ be its mutually exclusive receipt class. When $Q_t \neq \bot$, τ gives a matching successful resolution precedence. The receipt-transaction update is

$$(D^{\text{tx}}_{t+1}, Q_{t+1}) = \begin{cases} (D_t \oplus Q_t, \bot), & u_{t+1} = \text{resolved},\ Q_t \neq \bot, \\ (D_t, F_\Gamma(a_t; S_t)), & u_{t+1} = \text{clarify},\ Q_t = \bot, \\ (D_t \oplus F_\Gamma(a_t; S_t), Q_t), & u_{t+1} = \text{success}, \\ (D_t, Q_t), & \text{otherwise}. \end{cases} \tag{1}$$

A resolved receipt is serialized to its pending request. A fresh clarification cannot replace an existing pending effect, while an unrelated direct success preserves it. Idempotent typed assignment $\oplus$ makes duplicated observations harmless. The transactional state D^{tx}_{t+1} precedes receipt-backed graph mutation and deterministic closure, result, and phase recomputation into final (D_{t+1}, G_{t+1}). Destination-sensitive parsing gives terminal destinations precedence over source modifiers.

4.2 Reversible Evidence Update

Positive connection receipts add typed witness edges to G_t. Each edge records endpoint roles, object identity, resource identity, polarity, and receipt provenance. Derived closures retain the supporting edge set. A disconnection deletes every edge dependent on the disconnected object and clears downstream waiting and result registers. Closures require distinct resources and consistent device identities, which prevents evidence assembled from incompatible historical fragments.

4.3 Certificate Construction

The fixed schema Γ lists typed roles, receipt recognizers, effect templates, witness precedence, and action families that may satisfy each witness. It is frozen before confirmation evaluation under the protocol in Section 5.1.2 and contains no trajectory, answer, or instance-specific action sequence from any selected confirmation instance. The frontier retains interventions compatible with the earliest unmet witness together with navigation, observation, and interface-resolution actions. Result-dependent actions open only after a current evidence closure and waiting condition produce a typed result. For failure memory, ALFWorld sets $\Delta^M_\Gamma = 1$ on a persistent-key change or episode success, while ScienceWorld uses an official score increase. ScienceWorld exempts latent-time and epistemic families and budgets repeated epistemic queries separately. A recorded pair excludes only a configured action family at the same key. The six certificate classes address distinct error types.

The generic engine defines certificate and result types together with certified-subset projection and full-set fail-open. Environment-specific schemas instantiate typed roles, parsers, receipt recognizers, precedence, guards, revocation, and progress keys. Schema fidelity is audited on successful

reference paths through parsed families, topological frontiers, guard satisfaction, and memory availability. Projected retention is checked separately as the consequence predicted by Lemma 1. The authoring protocol and knowledge supplied to all compared methods are reported in Section 5.1.2.

4.4 Nonempty Action Projection

For every $a \in \mathcal{A}_t$, REGAP evaluates type, phase, frontier, evidence, result, and memory exclusion certificates. Let $\mathcal{J} = \{\text{type}, \text{phase}, \text{frontier}, \text{evidence}, \text{result}, \text{memory}\}$. For $j \in \mathcal{J}$, the binary indicator $X_j(a; S_t, \Gamma)$ equals one exactly when class j supplies a grounded certificate for excluding a; we suppress (S_t, Γ) below. The raw retained set is

$$\widetilde{\mathcal{A}}_t = \left\{ a \in \mathcal{A}_t : \sum_{j \in \mathcal{J}} X_j(a) = 0 \right\}. \tag{2}$$

REGAP returns $\widehat{\mathcal{A}}_t = \widetilde{\mathcal{A}}_t$ when this set is nonempty and returns $\widehat{\mathcal{A}}_t = \mathcal{A}_t$ otherwise. This fail-open case recovers complete-action prompting when every executable action has at least one exclusion certificate. A rule that lacks sufficient evidence abstains and contributes no exclusion.

Proposition 1 (Total executable interface)**.** *For every nonempty supplied set $\mathcal{A}_t$, REGAP returns $\varnothing \neq \widehat{\mathcal{A}}_t \subseteq \mathcal{A}_t$. If all certificate rules abstain or fail-open is triggered, $\widehat{\mathcal{A}}_t = \mathcal{A}_t$.*

Proof sketch. Certification only removes supplied actions. A nonempty certified subset is returned directly; otherwise fail-open returns the original nonempty set. Appendix Table B.3 compares explicit per-state set construction, receipt-gated commit, revocation, and nonempty-set guarantees in representative structured controllers. Marks follow the cited algorithmic formulations; shielding receives checks under its realizable safety-game assumptions.

4.5 Core Guarantees

Assumption 1 (Enumerated interaction interface)**.** *The supplied action set is nonempty and contains every executable action relevant to the claimed solution path. At most one clarification transaction is pending, and its responses are serialized to that request.*

Assumption 2 (Receipt and compiler fidelity)**.** *The evidence graph, pending effect, and failure memory are initialized as $G_0 = \varnothing$, $Q_0 = \bot$, and $M_0 = \varnothing$. The state-dependent adapter extraction, typed bindings, physical receipt recognizers, and result recognizers are faithful, meaning that they introduce no false state change and recognize every schema-governed transition used by the stated claims; success receipts have no false positives; and the effect compiler maps direct successes and serialized successful resolutions to the correct typed postconditions of their associated actions.*

Assumption 3 (Witness-schema fidelity)**.** *Consider a finite reference execution with a possible receipt sequence covered by Γ. The schema defines an acyclic precedence graph over its required witness families. At every raw interaction prefix, including a prefix awaiting clarification resolution, its parser maps the next reference action to a declared witness, support, or interface-resolution family. When that action advances a witness, the phase selects a minimal unmet witness in a topological order; the phase and frontier admit its family, including interface resolution while a transaction is pending; and the evidence and result guards accept the active typed prerequisites required by that transition. If the next action is memory-governed, that reference action has no zero-progress record at the current persistent key; an exempt next action produces no memory certificate. These conditions concern schema declarations, reference receipts, and persistent keys, independently of the output of P_Γ.*

Theorem 1 (Receipt-safe commit)**.** *Under Assumptions 1 and 2, every world-effect postcondition newly written to D^{tx}_{t+1} by Equation 1 has occurred in the environment. Each successful clarification transaction commits its pending effect exactly once, at its successful resolution step.*

Proof sketch. Direct commits require faithful success. A clarification stores an effect without changing the non-graph state, and only the matching successful resolution commits and clears it. Idempotent assignment makes repeated observations harmless.

Theorem 2 (Evidence revocation and result soundness)**.** *Under Assumption 2, every active edge in G_t has an unrevoked successful witness. Any exposed result is supported by a current type-consistent closure and a post-closure observation from the designated device.*

Proof sketch. Successful connections add witnessed edges, unrelated events preserve them, and disconnections remove every dependent edge. Result certificates conjoin current closure, role identity, resource distinction, waiting, and a faithful device observation.

Definition 3 (Step-wise corresponding path)**.** *After contracting each request, clarification, and successful resolution into one semantic macro action, two paths correspond when their ordered witness and support families plus task-relevant postconditions agree at every macro step.*

Lemma 1 (Local witness preservation)**.** *Under Assumptions 1–3, at every raw interaction prefix of the covered reference execution, its next reference action is supplied and executable, and all six exclusion indicators for that action are zero.*

Proof sketch. Enumeration supplies the next reference action. Parser fidelity defeats type exclusion, topological and transaction compatibility defeat phase and frontier exclusion, receipt-backed prerequisites defeat evidence and result exclusion, and the reference memory condition defeats memory exclusion.

Theorem 3 (Conditional path preservation)**.** *Under Assumptions 1–3, REGAP retains every action of the covered reference execution at its corresponding raw interaction prefix. Consequently, its retained action sets contain at least one step-wise corresponding path after clarification transactions are contracted.*

Proof sketch. Apply Lemma 1 and induct over raw interaction steps, including clarification resolutions. Receipt-safe commit advances verified postconditions, while revocation prevents an invalidated witness from enabling a later phase. Contracting completed transactions then gives the corresponding macro path.

Proposition 2 (Projection complexity)**.** *Let $n = |\mathcal{A}_t|$, let L bound action-text length, let H_t be the serialized size of the transition feedback inspected by the receipt recognizers, and let $s_t^C = \mathrm{size}(B_t, G_t, Q_t, Z_t)$ and $s_t^M = \mathrm{size}(M_t)$. With fixed schema Γ, bounded linear-time receipt recognizers, a bounded number of certificate rules per action, canonical action order supplied by the enumerator, and expected constant-time hashed memory lookup and insertion, the controller update and projection take expected $O\big(H_t + s_t^C + n(L + s_t^C)\big)$ time and $O(H_t + nL + s_t^C + s_t^M)$ total controller and working space. This bound excludes base-selector inference; projection adds no neural call. The time bound is linear in n when controller-state size is fixed. Caching state-derived closures and indexes can tighten it to $O(H_t + s_t^C + nL)$; if an adapter instead canonicalizes arbitrary action order by comparison sorting, this adds $O(n \log n\, L)$ comparison time.*

Proposition 1 and Theorems 1–3 give the core interface, state-integrity, and reachability guarantees under their stated assumptions. Proposition 2 characterizes controller cost. Appendix F gives auxiliary selector and certificate-risk analyses.

5 Experiments

We evaluate grounded interaction in ALFWorld and ScienceWorld through two research questions:

- **RQ1:** Does REGAP improve task completion and context efficiency across environments, model backbones, and planning or reasoning baselines?
- **RQ2:** Does the evidence-synchronized candidate interface account for the observed gains, and how do receipt gating, revocation, selector choice, and implementation affect those gains?

Experimental setup. We use deterministic decoding with Qwen2.5-7B-Instruct (Qwen Team, 2024) and replicate the final ALFWorld confirmation with Mistral-7B-Instruct-v0.3 (Jiang et al., 2023). A mechanism replication additionally uses QwQ-32B. ALFWorld permits at most 50 environment steps, and ScienceWorld 1.2.3 uses its `easy` configuration with 70 steps. Canonical model conditions use one fixed generation seed per paired task. The receipt/revocation, fixed-selector, and cardinality-matched diagnostics use five complete-run repeats, and the random-selector diagnostic uses ten. Decoding remains greedy, and complete-run repeats quantify residual variation in ScienceWorld execution. Comparisons are paired by task and model. Table 1 reports Holm-adjusted exact two-sided McNemar p-values over its eight pairwise comparisons, and we report Wilson intervals

for full-split completion; other measures are descriptive. Best and second-best completion within comparable groups are bold and underlined.

Baselines include the complete executable action set, an earlier Certificate Projection with Receipts (CPR) controller, ReAct and reflection adaptations (Yao et al., 2023b; Shinn et al., 2023), an MPO one-shot ReAct protocol, and an AdaPlanner protocol adaptation (Sun et al., 2023). CPR parses a simple goal and projects from receipt-confirmed holding, placement, and searched-location facts. REGAP generalizes this legacy controller to all six task families with typed phase schemas, pending effects, revocable evidence, bounded search coverage, and state-action failure memory. External methods are instantiated with the same frozen model and environment action contract. AdaPlanner retains its code prompt, task-family demonstrations, program execution, environment queries, assertion feedback, and repair loop.

5.1 RQ1: Completion and Efficiency Across Environments

5.1.1 Prospective Primary Comparison

We first transfer the unchanged schema and controller to twenty-four previously unused official-test variations across six ScienceWorld families, frozen before any selected reset, description, or outcome was accessed. Relative to complete actions, REGAP increases completion from 0/24 to 13/24, with thirteen REGAP-only successes, no reverse success, and Holm-adjusted exact two-sided $p = 1.71 \times 10^{-3}$. On a separate thirty-task official-train pool sealed before accessing any selected description or outcome, REGAP completes seventeen, versus one for complete actions and three each for ReAct-AC and ReflAct-AC; the three Holm-adjusted paired exact p-values are below 0.004, with 91.63 percent fewer input tokens than complete actions. On prospective conductivity tasks, REGAP completes thirteen of twenty-four versus two and retains all 470 supplied or syntax-resolved reference steps checked after performance. Across two disjoint ALFWorld confirmation sets sealed before their Qwen outcomes, it improves CPR from 96 to 108 with no reverse exclusive success. The Mistral comparison provides a cross-model replication on the balanced task set.

Table 1: Prospective confirmations and final-method replications across environments and model backbones.

Environment	Model	Comparator (success)	REGAP	Holm p
ScienceWorld held-out test	Qwen 7B	Complete actions (0/24)	**13/24**	1.71×10^{-3}
ScienceWorld held-out test	Qwen 7B	Full schema state (0/24)	**13/24**	1.71×10^{-3}
ScienceWorld multi-family	Qwen 7B	Complete actions (1/30)	**17/30**	1.16×10^{-3}
ScienceWorld multi-family	Qwen 7B	ReAct-AC (3/30)	**17/30**	2.44×10^{-3}
ScienceWorld multi-family	Qwen 7B	ReflAct-AC (3/30)	**17/30**	3.93×10^{-3}
ScienceWorld conductivity	Qwen 7B	Complete actions (2/24)	**13/24**	6.84×10^{-3}
ALFWorld disjoint	Qwen 7B	CPR (96/114)	**108/114**	2.44×10^{-3}
ALFWorld balanced	Mistral 7B	CPR (35/42)	**41/42**	3.13×10^{-2}

On the prospective multi-family comparison, projection reduces model input from 7,379,395 to 617,921 tokens, a 91.63 percent decrease, while using one model call per step rather than the two used by the reasoning adaptations. Each comparison uses the same sealed tasks and action contract.

5.1.2 OOD Planning Comparison

Table 2 compares REGAP with planning baselines on the full ALFWorld valid-unseen split. REGAP completes 131 of 134 tasks, exceeding AdaPlanner by 5.97 percentage points with ten-to-two discordance and MPO-plus with 45-to-one discordance. All table baselines were executed after method freeze.

Table 2: ALFWorld full-split comparison. Best in bold, second best underlined.

Method	Success	Rate (%)	Total tokens
ReAct-AC adaptation	50/134	37.31	3,946,762
ReflAct-AC adaptation	46/134	34.33	3,975,067
MPO-ReAct plus actions	87/134	64.93	23,096,203
AdaPlanner adaptation	<u>123/134</u>	<u>91.79</u>	1,437,032
REGAP	**131/134**	**97.76**	321,639

Against AdaPlanner on the same tasks, REGAP uses 77.62 percent fewer total tokens and 17.28 percent fewer environment steps. Appendix B reports model-call counts under the respective execution protocols. The largest family difference is on two-object placement, where REGAP completes 15 of 17 tasks and AdaPlanner completes 10.

Wilson intervals on the full split are 85.90 to 95.35 percent for AdaPlanner and 93.62 to 99.24 percent for REGAP. The paired comparison uses identical games and gives ten-to-two discordance with exact two-sided $p = 3.86 \times 10^{-2}$. Among the 121 tasks solved by both methods, REGAP reduces mean steps from 12.744 to 11.174 on the common success set.

Evidence and knowledge accounting. We use prospective confirmation sets fixed before evaluation, development-set analyses for component attribution, and a common-interface protocol for external methods.

At inference, REGAP receives no per-episode trajectory or demonstration and uses a frozen typed task-family schema. The evaluation unit is a previously unseen instance within a declared task family under that fixed specification. AdaPlanner and MPO each receive one matched example, whereas ReAct-AC and ReflAct-AC use two calls without demonstrations. Appendix Table B.5 reports the complete inference interface.

Answer to RQ1. Across the prospective ScienceWorld comparisons and the ALFWorld OOD comparison, REGAP improves completion over complete-action, reasoning, and planning baselines while reducing model input. The Mistral replication further preserves the improvement over CPR across a second model backbone.

5.2 RQ2: Mechanism and Component Attribution

5.2.1 Matched-Knowledge Schema Delivery

On the held-out official-test pool, the matched full-schema-state control exposes typed controller state, exact retained-action indices, and aggregate certificate facts while retaining complete actions; REGAP increases completion from 0/24 to 13/24. This prospective matched-knowledge comparison shows that compact projected candidate presentation delivers the controller state effectively in this protocol. On the earlier thirty-task pool, phase guidance and full schema state with complete actions each complete 1/30 tasks versus 17/30 for REGAP. With QwQ-32B, REGAP completes 14/30 tasks versus 0/30 for complete actions and reduces input tokens by 90.69 percent. Appendix Table B.7 reports the earlier controls.

Across five closed-loop repeats on thirty development tasks, equal-size hash-random subsets under the same model and decoder reach 0.8 ± 1.30/30 versus REGAP at 18.0 ± 0.00/30 and are lower every time; candidate count alone does not explain the gain (Appendix Table B.14).

5.2.2 Component Ablation

Performance remains stable without phase guidance at 19/24, while removing failure memory yields 13/24, evidence-only projection yields 4/24, and removing the frontier or using static projection yields 0/24. The frozen projection-only variant then reaches 13/24 versus 2/24 on disjoint confirmation tasks (Appendix Table B.4).

A repeated conductivity diagnostic directly intervenes on the two state-synchronization operations while holding the schema, frontier, selector, and decoding fixed. Across five repeats, the complete

controller has the highest completion in every repeat and averages 18.4 ± 0.55/24, compared with 14.2 ± 0.84/24 without receipt-gated commits, 15.0 ± 0.00/24 without revocation, and 14.0 ± 1.00/24 without both. Appendix Table B.8 reports the aggregate results.

The complete controller processes 118 disconnection events without retaining stale evidence. Removing receipt gating produces 40 unconfirmed commits, while removing revocation produces 150 retained stale-evidence instances.

5.2.3 Selector and Scaling Diagnostics

Selector controls separate projection from language choice. On the thirty-task ScienceWorld development pool, REGAP projection raises fixed-selector completion from 0.0/30 to 8.0/30 across five repeats, while language-conditioned selection reaches 17/30. Ten random-selector repeats further show that the projected interface remains effective under selector randomization in ALFWorld (Appendix Tables B.12 and B.13).

At 1,671 paired real states, projection reduces input from 534,928 to 336,649 tokens and the mean action set from 31.35 to 11.03. A fixed-state CPU benchmark over 256 to 2,048 actions has log-log slope 1.003 and is 21.62 times faster than the frozen implementation at the largest size, agreeing with Proposition 2. Exact replay over 3,484 states and bounded checks over 619 transitions satisfy every checked invariant; a 470-step gold-path audit verifies reference-path retention on all checked supplied or syntax-resolved steps. Appendix Table B.11 and Figure 2 give protocols and diagnostics, including fail-open counts.

Answer to RQ2. Matched-knowledge and cardinality controls attribute the gain to semantic projection rather than state prompting or candidate count. Component and selector interventions identify contributions from frontier control, receipt gating, revocation, and language-conditioned selection, while implementation diagnostics show context reduction and near-linear scaling.

6 Conclusion

REGAP turns receipt-confirmed, revocable evidence into nonempty projections for frozen language agents, improving completion and context efficiency across household and scientific interaction. Controls attribute gains to semantic projection, frontier control, receipt gating, revocation, and language selection. The analysis establishes totality, soundness, conditional path preservation, and efficient execution.

AI use statement

Generative AI tools assisted with research development, technical implementation, analysis, and manuscript preparation. The authors reviewed and verified all AI-assisted work and take responsibility for the final content of the paper.

Reproducibility statement

The appendix specifies benchmark splits, decoding, budgets, baseline adaptations, software, and hardware. Algorithm 1 and Appendix E provide pseudocode and complete proofs. The anonymous supplementary artifact contains frozen configurations, machine-readable summaries, source hashes, tests, and scripts needed to reproduce reported tables from the included results.

References

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *AAAI Conference on Artificial Intelligence*, 2018.

Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable reinforcement learning via policy extraction. In *Advances in Neural Information Processing Systems*, 2018.

Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. Prompting is programming: A query language for large language models. *Proceedings of the ACM on Programming Languages*, 7(PLDI): 1946–1969, 2023. doi: 10.1145/3591300.

Varun Pratap Bhardwaj. Agent behavioral contracts: Formal specification and runtime enforcement for reliable autonomous AI agents. *arXiv preprint arXiv:2602.22302*, 2026.

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*, 2026a.

Yaoqi Chen, Haibin Lai, Yuru Feng, Chuyu Han, Qianxi Zhang, Baotong Lu, Menghao Li, Xinjiang Wang, Zhirui Wang, Shusen Xu, Zengzhong Li, Zewen Jin, Hao Wu, Cheng Li, and Qi Chen. Beyond semantic organization: Memory as execution state management for long-horizon agents. *arXiv preprint arXiv:2606.06090*, 2026b.

Zheng Chen, Hanqing Liu, Duling Xu, Dong Dong, Jialin Li, Bangzheng Pu, and Jidong Zhai. Cordon: Semantic transactions for tool-using LLM agents. *arXiv preprint arXiv:2606.17573*, 2026c.

Javier García and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16:1437–1480, 2015.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. REALM: Retrieval-augmented language model pre-training. In *International Conference on Machine Learning*, 2020.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models. In *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 1769–1782. PMLR, 2023. URL `https://proceedings.mlr.press/v205/huang23c.html`.

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 *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 287–318. PMLR, 2023. URL `https://proceedings.mlr.press/v205/ichter23a.html`.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

Seongjae Kang, Taehyung Yu, and Sung Ju Hwang. PolicyGuide: From guarding one action to guiding the whole workflow for policy-compliant LLM agents. *arXiv preprint arXiv:2608.19861*, 2026.

Maurits Kaptein, Vassilis-Javed Khan, and Andriy Podstavnychy. Runtime governance for AI agents: Policies on paths. *arXiv preprint arXiv:2603.16586*, 2026.

Ehud Karpas, Omri Abend, Yonatan Belinkov, Barak Lenz, Opher Lieber, Nir Ratner, Yoav Shoham, Hofit Bata, Yoav Levine, Kevin Leyton-Brown, Dor Muhlgay, Noam Rozen, Erez Schwartz, Gal Shachaf, Shai Shalev-Shwartz, Amnon Shashua, and Moshe Tenenholtz. MRKL systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning. *arXiv preprint arXiv:2205.00445*, 2022.

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*, 2020.

Xiaoyang Li, Yiqi Wang, Haohui Lu, Zhi Chen, Mo Li, Pingan Song, Mingkai Zheng, and Taotao Cai. MemTX: Transactional belief commit for stateful agent memory. *arXiv preprint arXiv:2607.23929*, 2026.

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*, pp. 9493–9500, 2023. doi: 10.1109/ICRA48891.2023.10160591.

Shen Liu, Zhenguo Xu, Shaopu Wang, Yike Gao, and Chunlei Wang. HaReCAP: Habitual-action grounding for recursive large language model agents. *arXiv preprint arXiv:2608.16447*, 2026.

Gabriel Poesia, Oleksandr Polozov, Vu Le, Ashish Tiwari, Gustavo Soares, Christopher Meek, and Sumit Gulwani. Synchromesh: Reliable code generation from pre-trained language models. In *International Conference on Learning Representations*, 2022.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. *arXiv preprint arXiv:2307.16789*, 2023.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, 2023.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 9895–9901. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.779.

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*, 2023.

Basavesh Ammanaghatta Shivakumar, Swarn Priya, and Peng Gao. AgentFlow: A flow-centric policy language and framework for securing LLM agent systems. *arXiv preprint arXiv:2608.22868*, 2026.

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

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*, pp. 11523–11530, 2023. doi: 10.1109/ICRA48891.2023.10161317.

Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. AdaPlanner: Adaptive planning from feedback with language models. In *Advances in Neural Information Processing Systems*, 2023.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

Bhaskar Tripathi, Anurag Kumar, Ramendra Kumar, and Bhavesh Gadhe. A policy algebra for trust-preserving agentic AI execution. *arXiv preprint arXiv:2608.16402*, 2026.

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. *arXiv preprint arXiv:2305.16291*, 2023.

Haoyu Wang, Christopher M. Poskitt, and Jun Sun. AgentSpec: Customizable runtime enforcement for safe and reliable LLM agents. In *IEEE/ACM International Conference on Software Engineering*, pp. 2938–2950, 2026. doi: 10.1145/3744916.3764546.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. ScienceWorld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11279–11298, 2022. doi: 10.18653/v1/2022.emnlp-main.775.

Brandon T. Willard and Rémi Louf. Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*, 2023.

Jie Wu and Ming Gong. Policy-invisible violations in LLM-based agents. *arXiv preprint arXiv:2604.12177*, 2026.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, 2023a.

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*, 2023b.

Haiyu Zhao, Zhenyu Guo, Chunhong Zhang, Ziyu Zhou, and Zheng Hu. StructuThink: Reasoning with task transition knowledge for autonomous LLM-based agents. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 24489–24506, 2025. doi: 10.18653/v1/2025.findings-emnlp.1331.

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. *arXiv preprint arXiv:2310.04406*, 2023.

Yuqi Zhu, Shuofei Qiao, Yixin Ou, Shumin Deng, Shiwei Lyu, Yue Shen, Lei Liang, Jinjie Gu, Huajun Chen, and Ningyu Zhang. KnowAgent: Knowledge-augmented planning for LLM-based agents. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 3709–3732, 2025. doi: 10.18653/v1/2025.findings-naacl.205.

A Benchmark Datasets and Evaluation Protocol

A.1 ALFWorld

ALFWorld aligns text interaction with embodied household tasks (Shridhar et al., 2021). We use the TextWorld interface and its official out-of-distribution valid-unseen split containing 134 games. The six families are object placement, examination under light, cleaning then placement, heating then placement, cooling then placement, and placement of two objects. At each step the agent observes text and receives the environment's complete admissible action list. Success is the official binary task-completion signal. Runs use at most 50 steps, deterministic decoding, one fixed generation seed, and exact per-game pairing. The 42-game balanced confirmation and the later 72-game expansion are mutually disjoint official-train game files, jointly contain 19 games from each family, and form the sealed within-train confirmation pool. The full 134-game valid-unseen split is the OOD replication.

A.2 SCIENCEWORLD

ScienceWorld evaluates interactive scientific reasoning in a text environment with objects, devices, and compositional procedures (Wang et al., 2022). We use the ScienceWorld 1.2.3 `easy` configuration. The frozen study uses 24 previously unopened official-train conductivity variations that require preparation, construction of a valid circuit, observation, and result-dependent placement. The environment enumerates executable textual actions, including clarification responses. Completion is the official terminal success signal and score is the environment's cumulative progress score. Runs use at most 70 steps, deterministic decoding, one fixed generation seed, and paired tasks. Tasks are the independent analysis units for canonical model comparisons, with complete-run repeats used for stability diagnostics. A post-run audit over 24 paths and 470 supplied or syntax-resolved reference steps verifies retention of every checked reference action. This 24-task set is mutually disjoint from the 24-task development set used for Table B.4.

The prospective multi-family confirmation seals five official-train variations from each of six families before any selected description, reset, trajectory, outcome, or gold path is accessed. The families cover paint mixing, three object-classification tasks, plant identification, and conductivity. REGAP, complete actions, ReAct-AC, and ReflAct-AC use the same Qwen revision, seventy-step budget, deterministic decoding, and paired task order; reasoning adaptations receive no demonstrations or REGAP schema. Tasks serve as the independent evaluation units under deterministic decoding. The held-out transfer pool contains four previously unused official-test variations from each of the same six families. The controller, schema, prompt, model revision, decoding, and seventy-step budget were frozen before selected resets or descriptions were accessed; complete actions, REGAP, and the matched full-schema-state control use identical episodes and task order.

B PER-DATASET NUMERICAL RESULTS

Table B.3: Mechanism coverage in the cited formulations. Per-state set denotes explicit construction of an admissible subset; receipt commit denotes effect admission after environment confirmation; revocation denotes deletion of derived support; total nonempty denotes a stated nonempty-set guarantee.

Method	Per-state set	Receipt commit	Revocation	Total nonempty
KnowAgent (Zhu et al., 2025)	×	×	×	×
StructuThink (Zhao et al., 2025)	×	×	×	×
AgentSpec (Wang et al., 2026)	×	×	×	×
Shielding (Alshiekh et al., 2018)	✓	×	×	✓
REGAP	✓	✓	✓	✓

Table B.4: ScienceWorld development-set component analysis.

Condition	Completion
Receipt-aware projection with guidance	14/24
Projection without guidance	**19/24**
No failure memory	13/24
Evidence projection without phase state	4/24
No feasible frontier	0/24
Static schema projection	0/24

Table B.5: Domain-knowledge and inference interface. Demo denotes examples supplied per episode.

Method	Demo	Task-family specification	Calls/step
Complete actions	0	no	1
ReAct-AC / ReflAct-AC	0	no	2
MPO-ReAct	1	no	1
AdaPlanner adaptation	1	program prompt	adaptive
REGAP	0	typed adapter	1

The sealed ScienceWorld primary pool spans six task families and eleven observed action families. The authoring protocol permits public interfaces, declared task families, action templates, and explicitly named development data before freeze, while selected confirmation descriptions, trajectories, scores, outcomes, and gold paths remain sealed.

Table B.6: Prospective multi-family ScienceWorld comparison on the sealed thirty-task pool.

Method	Success	Mean score	Mean steps	Input tokens	Calls
Complete actions	1/30	-1.70	57.07	7,379,395	1,712
ReAct-AC adaptation	3/30	11.27	45.93	15,828,433	2,756
ReflAct-AC adaptation	3/30	-12.97	36.30	11,379,480	2,178
REGAP	**17/30**	83.67	36.60	617,921	1,098

Table B.7: Development-set mechanism controls retain complete actions while exposing phase guidance and typed schema state.

Model	Condition	Success	Input tokens
Qwen2.5-7B	Phase guidance, complete actions	1/30	4,999,669
Qwen2.5-7B	Full schema state, complete actions	1/30	6,091,966
Qwen2.5-7B	REGAP	**17/30**	**617,921**
QwQ-32B	Complete actions	0/30	7,805,103
QwQ-32B	REGAP	**14/30**	**726,500**

Table B.8: Receipt and revocation diagnostic over five repeats on the ScienceWorld conductivity development pool. Values are mean completion $\pm$ sample standard deviation.

Condition	Mean completion
Complete controller	$\mathbf{18.4 \pm 0.55}$/24
No receipt-gated commit	14.2 ± 0.84/24
No revocation	15.0 ± 0.00/24
No receipt-gated commit or revocation	14.0 ± 1.00/24

Table B.9: ALFWorld task-family completion. Best in bold, second best underlined.

Family	Games	AdaPlanner	REGAP
Place	24	**24**	**24**
Examine	18	17	**18**
Clean	31	29	**30**
Heat	23	**23**	**23**
Cool	21	20	**21**
Place two	17	10	**15**

Table B.10: Efficiency on the full ALFWorld split. Best in bold and second best underlined when directly comparable.

Measure	AdaPlanner	REGAP
Input tokens	974,351	**311,671**
Output tokens	462,681	**9,968**
Total tokens	1,437,032	**321,639**
Environment steps	1,979	**1,637**
Model calls	836	1,637
Common-success steps	12.744	**11.174**

Table B.11: Token reductions under distinct, explicitly named comparison units.

Comparison and unit	Comparator	REGAP	Reduction
ALFWorld vs AdaPlanner, total	1,437,032	321,639	77.62%
ScienceWorld multi-family, input	7,379,395	617,921	91.63%
ScienceWorld conductivity, input	14,308,842	614,001	95.71%
ScienceWorld held-out test, input	10,752,361	938,730	91.27%
Paired prompt reconstruction, input	534,928	336,649	37.07%

Table B.12: Random-selector attribution over ten repeats.

Environment	Random full	Random REGAP	Frozen Qwen reference
ALFWorld valid-unseen	5/1,340	1,310/1,340	REGAP 131/134
ScienceWorld frozen	0/240	2/240	Full 2/24; REGAP 13/24

Across ten random-selector repeats, the completion totals separate the contribution of projection from selector choice, and canonical deterministic reruns reproduce every per-episode completion indicator. The projected ALFWorld interface averages 10.16 actions and is a singleton on 38.44 percent of steps. On ScienceWorld, projection and language-conditioned choice provide complementary contributions.

Table B.13: Fixed-selector attribution over five repeats on the thirty-task ScienceWorld development pool; frozen Qwen is the canonical reference. Repeated-condition values are mean completion $\pm$ sample standard deviation.

Selector and interface	Completion
Lexicographic fixed, full actions	0.0 ± 0.00/30
Lexicographic fixed, REGAP projection	8.0 ± 0.00/30
Frozen Qwen, REGAP projection	**17/30**

Within the fixed-selector control, the selector is held constant while the presented candidate set changes. Its intermediate completion separates the value of projection from the additional value supplied by language-conditioned choice.

Table B.14: Closed-loop cardinality-matched control over five repeats on the thirty-task ScienceWorld development pool. At every visited state, the hash-random subset contains the same number of actions as REGAP's semantic projection. Values are mean completion $\pm$ sample standard deviation.

Candidate interface	Completion
Cardinality-matched hash-random subset	0.8 ± 1.30/30
REGAP semantic projection	$\mathbf{18.0 \pm 0.00}$/30

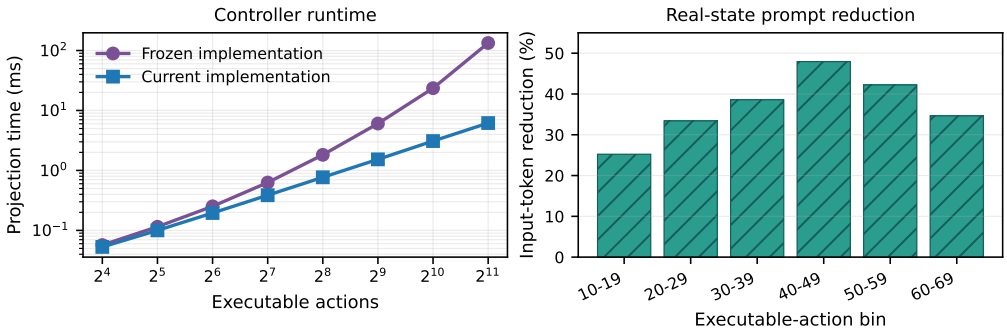

Figure 2: The projection-only CPU benchmark at fixed controller state and paired prompt reduction as executable action sets grow.

C Implementation Details

C.1 Our Method's Hyperparameters

REGAP uses no learned parameters or task-specific numerical threshold search. Qwen experiments use Qwen2.5-7B-Instruct revision `a09a35458c70`; the complete revision hash is recorded in the artifact. The ScienceWorld mechanism replication uses QwQ-32B revision `976055f8c83f`, and the cross-model replication uses Mistral-7B-Instruct-v0.3. Decoding uses temperature zero and one fixed generation seed. ScienceWorld action selection uses exact-action prefix constraints for every backbone, including QwQ-32B. ALFWorld uses batch size 16, at most 32 new tokens per action, and 50 environment steps. The prospective ScienceWorld conductivity and six-family studies use 70 environment steps. The evidence-closure waiting gate is two observations and clarification capacity is one pending transaction. For failure memory, ALFWorld hashes persistent executable and certified fields, while ScienceWorld hashes persistent look, inventory, and score; neither key includes the memory itself. All prospective choices were fixed in protocol manifests before their respective confirmation outcomes, and mechanism replications are labeled separately. The receipt/revocation diagnostic crosses the two update operations over five repeats, and the fixed-selector diagnostic uses the same lexicographic selector with full and projected candidate sets over five repeats. The cardinality-matched control uses the same model, prompt structure, greedy decoder, and tasks; at every visited state, content-agnostic hash ranking selects from the full set exactly as many actions as REGAP retains.

C.2 Baseline Hyperparameters

The complete-action baseline shares every model, tokenizer, decoding setting, step budget, and prompt field with REGAP, replacing only the projected set by the full executable set. The phase-guidance control retains the full executable set and adds the controller-derived phase guidance used by the earlier guidance-bearing condition. CPR shares the complete-action settings and uses its frozen legacy projection implementation based on receipt-confirmed goal, holding, placement, and searched-location facts. REGAP extends this controller with six-family typed phase schemas, pending-effect state, revocable graph evidence, bounded coverage reopening, and state-action failure memory. ReAct-AC and ReflAct-AC use the same Qwen checkpoint, temperature zero, complete admissible-action contract, and no demonstrations. Each uses one concise thought or reflection call followed by one constrained action-selection call per step, with a 50-step ALFWorld budget and 70-step ScienceWorld budget. MPO uses commit `5529eca`, one demonstration for each task family, one joint thought-and-action call, its official instruction, and complete conversation history. MPO-plus additionally receives the current admissible-action list. AdaPlanner uses official code prompts, one example per family, at most six program attempts, a 50-step budget, assertion-triggered feedback, three resume repairs, three restart repairs, and a safety-checked Python subset. Its pinned repository commit is `dba187601500e7d91fee5fd786a455aaa7988677`.

Table C.15: Generation budgets and information supplied to each final comparison.

Method	Output cap	Calls	Demo	Frozen configuration
REGAP / complete	exact action or 32 tokens	1	0	REGAP full split and confirmation
CPR	32 tokens	1	0	REGAP full split
ReAct-AC / ReflAct-AC	48 tokens plus exact action	2	0	reasoning-baseline configs
MPO-ReAct	128 tokens	1	1	MPO full-split config
AdaPlanner	1,400 tokens per generation	adaptive	1	AdaPlanner full-split config

The artifact retains the complete configuration names, prompt hashes, model revisions, generation configuration, task ordering, and source hashes. Each frozen ScienceWorld reasoning configuration records the prompt-module hash, and every exact-action call is capped by the longest tokenized admissible action plus the end token rather than an arbitrary generation limit. The canonical machine-readable summaries link each reported number to its result file and hash.

C.3 REGAP Algorithm

Let T denote the episode step budget. The other inputs are schema Γ, frozen selector π, and the environment interface.

Algorithm 1 Reversible Evidence-Graph Action Projection

Receive $(o_0, \mathcal{A}_0)$; initialize (B_0, Z_0) from Γ and set $G_0 = \varnothing$, $Q_0 = \bot$, $M_0 = \varnothing$, r_0 null, and $t = 0$
while $t < T$ and the episode is nonterminal **do**
  Evaluate $X_j(a; S_t, \Gamma)$ for every $a \in \mathcal{A}_t$ and $j \in \mathcal{J}$
  Form $\widetilde{\mathcal{A}}_t$ by Equation 2; set $\widehat{\mathcal{A}}_t = \widetilde{\mathcal{A}}_t$ if nonempty and $\mathcal{A}_t$ otherwise
  Select and execute $a_t \sim \pi(o_t, \widehat{\mathcal{A}}_t)$
  Receive o_{t+1} and terminal flag d_{t+1}, plus $\mathcal{A}_{t+1}$ if $d_{t+1} = 0$
  Derive $r_{t+1} = R_\Gamma(S_t, o_t, a_t, o_{t+1})$ and classify $u_{t+1} = \tau(r_{t+1})$
  Update $(D^{\mathrm{tx}}_{t+1}, Q_{t+1})$ by Equation 1
  Update G_{t+1} from r_{t+1}; deterministically derive D_{t+1} from D^{tx}_{t+1}, current closures, waiting gates, results, and phases
  Set $M_{t+1} = M_t \cup \{(\kappa_t, a_t)\}$ if $\Delta^M_\Gamma(S_t, a_t, o_{t+1}) = 0$ and set $M_{t+1} = M_t$ otherwise; enable matching memory exclusion only for schema-configured families
  Set $t \leftarrow t + 1$
end while

D Computational Resources

Experiments ran on NVIDIA A100-SXM4 80 GB GPUs using Python 3.11.15, PyTorch 2.5.1 with CUDA 12.1, Transformers 4.51.3, ALFWorld 0.4.2, TextWorld 1.7.0, and ScienceWorld 1.2.3.

E Theorem Proofs

E.1 Proof of Proposition 1

Proof. Every certificate is attached to an action already in $\mathcal{A}_t$, so certification cannot add an external action. If at least one action has no exclusion certificate, the returned certified set is a nonempty subset of $\mathcal{A}_t$. Otherwise fail-open returns $\mathcal{A}_t$. When every rule abstains, no action receives an exclusion certificate and the certified set equals $\mathcal{A}_t$. The result uses no receipt, compiler, schema, or selector assumption. □

E.2 Proof of Theorem 1

Proof. Consider the transition from t to $t+1$. When $Q_t \neq \bot$, a matching successful resolution is classified as resolved. By Assumption 1, that receipt is serialized to its pending request, and Assumption 2 establishes every postcondition in Q_t before commit. The update then clears Q_t. A fresh clarification stores $F_\Gamma(a_t; S_t)$ in Q_{t+1} without changing D_t. A direct-success receipt has no false positive, so compiler fidelity establishes every postcondition in $F_\Gamma(a_t; S_t)$ before commit, while the update preserves any unrelated pending effect. A fresh clarification is stored only when no transaction is already pending. The classes are mutually exclusive, no other case writes a world-effect postcondition to D^{tx}_{t+1} through Equation 1, and typed assignment is idempotent. Later graph and deterministic derived-state updates are outside the transaction claim. Hence each successful clarification transaction commits its pending effect exactly once. □

E.3 Proof of Theorem 2

Proof. Induct on interaction steps. The initial graph is empty. Under Assumption 2, a successful connection receipt adds an edge whose endpoints, roles, polarity, resource, and provenance describe an occurred transition. Unrelated events preserve the invariant. A disconnection removes every dependent edge, so no revoked witness remains active. A result certificate requires a current closure with the designated source, device, specimen, distinct resources, consistent polarities, a post-closure wait, and a faithful device observation. Each conjunct follows from an active witness or current observation. □

E.4 Proof of Lemma 1

Proof. Fix a raw interaction prefix of the covered reference execution, including any prefix at which a clarification transaction awaits resolution, and let $a^\star_t$ be its next reference action. Assumption 1 places $a^\star_t$ in $\mathcal{A}_t$. Under Assumption 3, the parser maps $a^\star_t$ to its declared family, so its type indicator is zero. The phase and frontier admit that family, including interface resolution while pending, so their indicators are zero. Assumption 2 makes the reference prefix's typed prerequisites current, while guard fidelity makes the evidence and result indicators zero. The reference memory condition or configured family exemption makes the memory indicator zero. Hence $X_j(a^\star_t; S_t, \Gamma) = 0$ for every $j \in \mathcal{J}$. □

E.5 Proof of Theorem 3

Proof. Fix the possible environment and receipt realization of the finite reference execution from Assumption 3, and induct over its raw interaction steps. At each reference prefix, Lemma 1 gives the exact next reference action $a^\star_t$ with every exclusion indicator equal to zero. Equation 2 therefore places $a^\star_t$ in $\widetilde{\mathcal{A}}_t$ and hence in $\widehat{\mathcal{A}}_t$. Following that retained action and the fixed possible receipt realization produces the next raw reference prefix: clarification requests first update Q_t, their resolution actions are covered by the next induction step, receipt-safe commit advances verified postconditions, and evidence soundness maintains active graph consequences. Thus every raw reference action is retained. Contracting each completed request, clarification, and successful resolution as in Definition 3 yields a path with the same ordered witness and support families plus task-relevant postconditions. □

E.6 Proof of Proposition 2

Proof. Receipt and observation parsing costs $O(H_t)$ under the bounded linear-time recognizer set, and controller-state updates scan at most s^C_t stored non-memory content. Each of the n actions has length at most L and triggers a bounded number of certificate rules. Parsing an action costs $O(L)$, and an uncached state-dependent rule can inspect at most s^C_t controller content, giving expected $O(H_t + s^C_t + n(L + s^C_t))$ time when hashed memory lookup and insertion have expected constant cost. If state-derived closures and membership indexes are computed once after each receipt and then indexed during certification, the state term is paid once and the bound tightens to $O(H_t + s^C_t + nL)$. Transition parsing, candidate storage, non-memory controller state, and memory require $O(H_t + nL + s^C_t + s^M_t)$ total space. Base-selector inference is outside these controller bounds, and

projection invokes no additional neural call. The stated bound assumes canonical order is supplied; comparison sorting used to canonicalize arbitrary action order adds $O(n \log n\, L)$ time. □

F Auxiliary Analysis

F.1 Selector Value Decomposition

Assumption 4 (Selector conditioning)**.** *For the value analysis, the base selector gives the returned set positive support, $\pi_{\mathcal{A}_t}(\widehat{\mathcal{A}}_t) > 0$. Supplying any $U \subseteq \mathcal{A}_t$ with $\pi_{\mathcal{A}_t}(U) > 0$ conditions that selector: $\pi_U(a) = \pi_{\mathcal{A}_t}(a)/\pi_{\mathcal{A}_t}(U)$ for $a \in U$. The uniform-selector diagnostic realizes this relation exactly, and the analysis uses it to model re-prompted language selection.*

Theorem 4 (Selector value decomposition)**.** *Let $q_t(a)$ be bounded continuation value under a common downstream policy. Define $V_t^{\mathrm{full}} = \mathbb{E}_{a\sim\pi_{\mathcal{A}_t}}[q_t(a)]$, $V_t^{\mathrm{REGAP}} = \mathbb{E}_{a\sim\pi_{\widehat{\mathcal{A}}_t}}[q_t(a)]$, and $\rho_t = \pi_{\mathcal{A}_t}(\mathcal{A}_t \setminus \widehat{\mathcal{A}}_t) \in [0, 1)$. The conditional expectations below are taken under $a \sim \pi_{\mathcal{A}_t}$. Under Assumption 4, $\rho_t = 0$ implies $V_t^{\mathrm{REGAP}} = V_t^{\mathrm{full}}$. If $\rho_t > 0$, then*

$$V_t^{\mathrm{REGAP}} - V_t^{\mathrm{full}} = \rho_t \left(\mathbb{E}[q_t(a) \mid a \in \widehat{\mathcal{A}}_t] - \mathbb{E}[q_t(a) \mid a \notin \widehat{\mathcal{A}}_t] \right). \tag{3}$$

Thus, for positive removed mass, projection improves one-step selector value exactly when retained actions have larger base-policy-weighted mean continuation value.

Proof. Write $K = \widehat{\mathcal{A}}_t$, $K^{\mathrm{c}} = \mathcal{A}_t \setminus K$, and $\rho_t = \pi_{\mathcal{A}_t}(K^{\mathrm{c}})$. Positive support from Assumption 4 gives $\pi_{\mathcal{A}_t}(K) = 1 - \rho_t > 0$, hence $\rho_t < 1$ and the conditioned selector is defined. If $\rho_t = 0$, conditioning on K removes zero probability mass, so the two values agree. If $\rho_t > 0$, Assumption 4 gives $V_t^{\mathrm{REGAP}} = \mathbb{E}[q_t(a) \mid a \in K]$, while total expectation gives $V_t^{\mathrm{full}} = (1 - \rho_t)\mathbb{E}[q_t(a) \mid a \in K] + \rho_t \mathbb{E}[q_t(a) \mid a \in K^{\mathrm{c}}]$. Subtraction proves Equation 3. □

F.2 Trajectory Bound with Fallible Certificates

Theorem 5 (Trajectory bound with fallible certificates)**.** *For a fixed-horizon common coupling of task, environment, and selector randomness, let J_{REGAP} and J_{full} be returns in a common interval of width R, and suppose at most m exclusion certificates are selected along the REGAP trajectory. Let $\mathcal{E}_i$ be the event that the ith selected certificate is unsound, with $\mathcal{E}_i = \varnothing$ when fewer than i certificates are selected, and suppose $\Pr(\mathcal{E}_i) \le \epsilon_i$. If, whenever $\Pr(\mathcal{H}) > 0$, the all-sound event $\mathcal{H} = \bigcap_{i=1}^{m} \mathcal{E}_i^c$ satisfies $\mathbb{E}[J_{\mathrm{REGAP}} - J_{\mathrm{full}} \mid \mathcal{H}] \ge g \ge 0$, then without event independence*

$$\mathbb{E}[J_{\mathrm{REGAP}} - J_{\mathrm{full}}] \ge (1 - \delta)g - \delta R, \qquad \delta = \min\left\{1, \sum_{i=1}^{m} \epsilon_i\right\}. \tag{4}$$

Proof. The complement of the all-sound event satisfies $\mathcal{H}^c = \bigcup_{i=1}^{m} \mathcal{E}_i$, so the union bound gives $\Pr(\mathcal{H}^c) \le \delta$ without independence. If $\Pr(\mathcal{H}) = 0$, this inequality forces $\delta = 1$, and bounded return proves Equation 4. Otherwise, conditional gain on $\mathcal{H}$ is at least g, while gain on $\mathcal{H}^c$ is at least $-R$; conditioning and the same probability bound prove the result. □

G Limitations

REGAP provides a modular interface between environment feedback and action selection. Future work can study automated construction of its typed interface and extend the same evidence-synchronization mechanism to multimodal and open-ended tool settings.