AFTER DENIAL, WHAT NEXT? RECOVERABILITY-AWARE LEGAL CONTINUATION FOR TOOL-USING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tool-using agents pursue user goals through calls to external services. Authorization determines which tool actions are permitted, but a denial leaves an unresolved decision: which permitted action should the agent take next? A poor continuation can stall a recoverable task or request unnecessary capability. Existing authorization checks enforce the legal boundary without ranking the ways forward. We formulate *post-denial legal continuation* and introduce Denial-Aware Recovery Projection (DARP). Given structured denial feedback, DARP constructs typed recovery candidates, checks them against the current policy, and estimates their bounded-horizon task value. It selects among near-maximal legal candidates by capability exposure, execution cost, and repair distance, then binds the chosen action to a one-time state-bound certificate for dispatch. On 320 template-held-out tasks with registered deterministic transitions, exact values, a three-step horizon, and complete candidate recall, DARP achieves 100% safe resolution with 27% less mean capability exposure than a bounded-horizon legal value filter. Replacing exact values with a frozen Qwen scorer yields 90.94% safe resolution with no unauthorized dispatch.

1 INTRODUCTION

Once a language agent can call external tools, its decisions can affect more than the text it produces. ReAct interleaves reasoning with actions, and Toolformer learns to call APIs (Yao et al., 2022; Schick et al., 2023). Such calls may read private data or change external state, making authorization a necessary boundary (Shi et al., 2025; Debenedetti et al., 2025). Rejecting a disallowed call preserves that boundary, but it does not necessarily resolve the user's task.

Progent checks calls against privilege policies (Shi et al., 2025), CaMeL tracks capabilities through execution (Debenedetti et al., 2025), and MiniScope derives least-privilege permissions from tool-scope relations (Zhu et al., 2025). These mechanisms establish what the agent may do. Progent's email-to-Slack example makes the remaining decision concrete: after an injected email call is blocked, the agent can still deliver the authorized summary through Slack (Shi et al., 2025). Other denials may leave several permitted next steps, differing in both task progress and capability use. Authorization determines which actions are permitted; post-denial recovery determines which permitted action is worth taking.

We formulate this choice as *post-denial legal continuation*. Given an authenticated objective, trusted state, policy, held capabilities, and a denied proposal, Denial-Aware Recovery Projection (DARP) constructs typed alternatives from the failed authorization predicates. The deployed checker filters these alternatives, and bounded search estimates the recoverable value of each legal first action. DARP retains actions near the best estimated value, then selects by requested capability exposure, direct capability use, execution cost, and repair distance. The selected action is rechecked at dispatch under a one-time certificate bound to its arguments, state, policy, and tool interface. Its observed return updates the state before the agent replans. Figure 1 shows this decision loop.

The design separates authorization from the quality of the recovery choice. Sound checking and atomic certified dispatch establish current-state legality; with uniformly bounded value error, the selected first action has candidate-relative regret at most $\epsilon + 2\delta$. We evaluate the choice on post-

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.

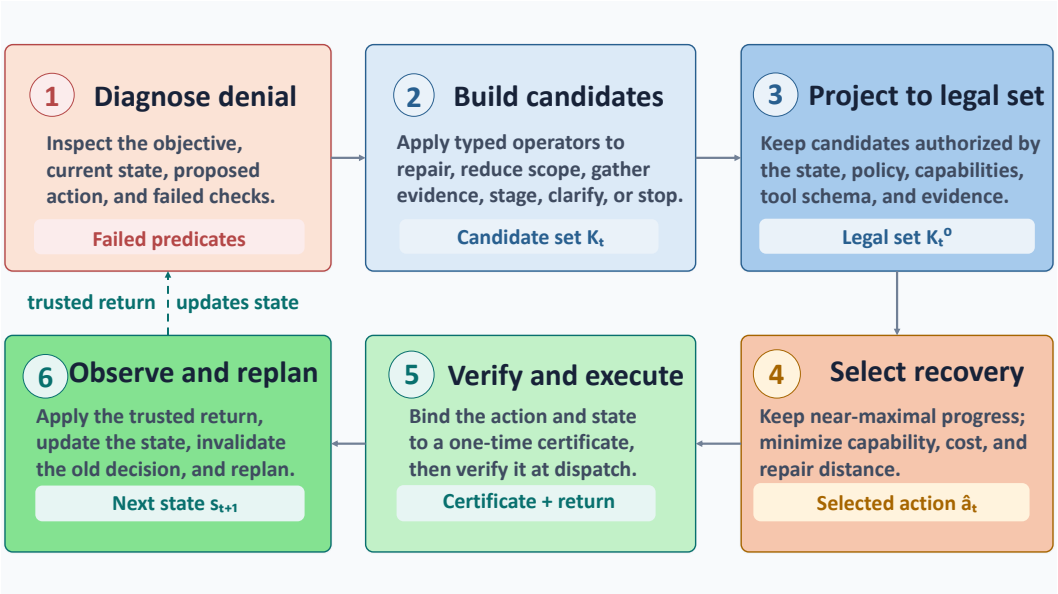


Figure 1: Overview of DARP. Structured denial feedback produces typed candidate continuations. The deployed checker defines the current legal set, bounded search identifies a near-maximal recovery frontier, and the lexicographic selector chooses a low-exposure continuation from that frontier. A state-bound certificate mediates dispatch; the observed return updates the state before the next plan.

denial Gmail, Drive, Calendar, and cross-application workflows. On 320 template-held-out tasks with registered deterministic transitions, exact values, a three-step horizon, and complete candidate recall, DARP and a bounded-horizon legal value filter both achieve 100% safe resolution, while DARP reduces mean capability exposure from 10.20 to 7.44. Replacing exact values with a frozen Qwen scorer yields 90.94% safe resolution and zero unauthorized dispatch. The contrast shows why selecting a useful legal action and enforcing its legality must be assessed separately.

2 Related Work

Least privilege and complete mediation provide the classical basis for controlling access to protected resources (Saltzer & Schroeder, 1975), while runtime enforcement characterizes properties that a monitor can guarantee over an action sequence (Schneider, 2000). Recent agent systems instantiate these principles at the language-model–tool boundary. CaMeL separates trusted control flow from untrusted data and propagates capabilities through execution (Debenedetti et al., 2025); Progent mediates calls with privilege policies (Shi et al., 2025); and MiniScope derives task-specific permissions from tool-scope relations (Zhu et al., 2025). IsolateGPT isolates application execution (Wu et al., 2024), while Task Shield checks whether calls serve the user's task (Jia et al., 2025). ToolSafe supplies step-level feedback (Mou et al., 2026), and VIGIL verifies tool-stream actions before commit (Lin et al., 2026). These systems clarify the execution boundary on which post-denial recovery depends: a checker determines feasibility, while the remaining choice is among feasible continuations.

Failure feedback also supports revision and planning. Reflexion uses verbal feedback across trials (Shinn et al., 2023), Self-Refine iterates critique and rewriting (Madaan et al., 2023), and CRITIC grounds correction in tool feedback (Gou et al., 2024). A survey of self-correction emphasizes the importance of informative feedback for reliable revision (Kamoi et al., 2024). AdaPlanner adapts plans to environment feedback (Sun et al., 2023); Tree of Thoughts and Language Agent Tree Search explore value-guided intermediate states (Yao et al., 2023; Zhou et al., 2023). ToolMaze examines replanning under tool failures (Zhu et al., 2026). An authorization denial gives revision a more specific starting point: failed capability, scope, provenance, confirmation, or state predicates. The revised actions must still pass the current checker before their recoverable value can be compared.

Constrained decision methods likewise separate feasible actions from the choice among them. Shielding restricts the actions available to a policy (Alshiekh et al., 2018), and predictive safety filters modify proposed controls to satisfy constraints (Wabersich & Zeilinger, 2021). In our setting, the checker defines the legal set of typed tool and control calls; bounded-horizon value and semantic capability costs determine the preferred continuation inside that set. Evaluation therefore needs to measure both the outcome of recovery and the capability used to obtain it.

Existing tool-use evaluations cover large API collections, including Gorilla and ToolLLM (Patil et al., 2024; Qin et al., 2024), and simulated or tool-augmented calls in ToolAlpaca and API-Bank (Tang et al., 2023; Li et al., 2023). Other studies measure tool manipulation or function calling (Xu et al., 2023; Patil et al., 2025); AgentBench, ToolSandbox, and τ-bench include agent interaction and stateful tasks (Liu et al., 2024; Lu et al., 2025; Yao et al., 2024). Safety evaluations examine risky tool use (Ruan et al., 2024; Ye et al., 2024) and indirect prompt injection (Zhan et al., 2024; Debenedetti et al., 2024). Agent Security Bench studies attacks and defenses (Zhang et al., 2025); AgentPoison studies memory poisoning (Chen et al., 2024); and Greshake et al. (2023) demonstrate indirect injection in integrated applications. Our benchmark starts after a structured denial and measures the legality, task value, and capability footprint of the ensuing choice, followed by its integrity at dispatch.

3 METHOD

After a tool action is denied, recovery starts from the action that failed, the current authorization state, and the user's objective. Several legal next steps may remain, with different consequences for progress and capability use.

Problem formulation and legal continuation. Let q be an authenticated user objective, $s_t \in \mathcal{S}$ the trusted environment state, P_t the current policy, and C_t the held provider scopes or direct capabilities. A typed action is

$$a = \big(\mathrm{op}(a), \mathrm{arg}(a), \mathrm{pre}(a), \mathrm{eff}(a), \mathrm{Cap}(a), \mathrm{Req}(a)\big), \tag{1}$$

where op and arg identify the call and its arguments, pre and eff describe its registered preconditions and effects, and $\mathrm{Cap}(a)$ is the semantic capability needed to execute a. For an authorization-request control action, $\mathrm{Req}(a)$ contains the scopes being requested, while $\mathrm{Cap}(a)$ contains the already-held capability needed to issue that request. Ordinary tool actions have $\mathrm{Req}(a) = \varnothing$.

An external checker returns $\mathrm{Adm}(a, s_t, P_t) \in \{0, 1\}$ and exposes a coverage function $\mathrm{Cov}(C_t)$ from provider scopes to semantic capabilities. If the proposed action a_t^- is denied, the trusted recovery record is

$$d_t = \big(a_t^-, \tau_t, F_t, \nu(s_t), \nu(P_t)\big), \tag{2}$$

where τ_t is a normalized denial type, F_t is the set of failed predicates, and $\nu(\cdot)$ records versions. A generator Γ applies a registry of typed recovery operators Ω to obtain a finite set

$$K_t = \Gamma(q, s_t, d_t; \Omega). \tag{3}$$

Let $\mathcal{R}_t$ denote the provider scopes that the current policy permits the agent to request. The legal continuation set is

$$\begin{aligned} K_t^0 = \{a \in K_t :\ & \mathrm{Adm}(a, s_t, P_t) = 1, \\ & \mathrm{Cap}(a) \subseteq \mathrm{Cov}(C_t), \\ & \mathrm{Req}(a) \subseteq \mathcal{R}_t\}. \end{aligned} \tag{4}$$

The request condition is vacuous for ordinary actions. A request action does not add a scope by itself; C_{t+1} changes only after a trusted approval event. A side-effect-free stop action $a_\perp$ is registered as admissible under every policy, with empty capability and request sets and zero cost; thus K_t^0 is nonempty.

For legal continuations, let $U_q(s) \in [0, 1]$ measure verified task progress. Each action has the schema-derived cost

$$c(a) = \tfrac{1}{4}\big(z_{\mathrm{tool}} + z_{\mathrm{user}} + z_{\mathrm{write}} + z_{\mathrm{irrev}}\big) \in [0, 1], \tag{5}$$

where the four binary attributes indicate a tool call, user interruption, persistent write, and irreversible side effect. These attributes are frozen per registered action type before evaluation. Let $\eta \geq 0$ be the fixed cost coefficient. The true legal recovery value of first action a over horizon H is

$$V_H(s_t, a; q) = \max_{\pi \in \Pi_H^0(s_t, a)} \mathbb{E}_\pi \left[U_q(s_{t+H}) - \eta \sum_{j=t}^{t+H-1} c(a_j) \right]. \tag{6}$$

The policies in $\Pi_H^0(s_t, a)$ begin with a and use only actions legal in their respective execution states; a stop action leaves the state unchanged for the remaining horizon. Let a_t^{opt} maximize $V_H(s_t, a; q)$ over K_t^0. The candidate-relative recovery regret of a selected action a_t^* is

$$\mathrm{Reg}_H(a_t^*) = V_H(s_t, a_t^{\mathrm{opt}}; q) - V_H(s_t, a_t^*; q). \tag{7}$$

Typed recovery and bounded search. The denial type belongs to

$$\begin{aligned} \tau_t \in \{&\textsc{MissingCap}, \textsc{ScopeMismatch}, \textsc{FailedPrecond}, \\ &\textsc{UntrustedArg}, \textsc{NeedsConfirm}, \textsc{Other}\}. \end{aligned} \tag{8}$$

The six operator families address these failures in different ways. Argument repair replaces malformed or untraceable values with trusted ones, while scope reduction narrows resources or effects. Evidence acquisition establishes a failed precondition; clarification obtains a missing value, confirmation, or permitted authorization. Reversible staging creates a draft, backup, or preview, and termination reports that no useful legal continuation is available. Each operator is a partial typed transformation

$$\omega : (q, s_t, d_t) \mapsto \{(a, \mathrm{pre}_\omega, \mathrm{eff}_\omega)\}. \tag{9}$$

Arguments are schema-validated, semantic duplicates are canonicalized, and a non-repeatable action whose modeled transition leaves the state unchanged is removed. Staging is represented as a real transition: the stage action establishes `staged_ready`, which is a precondition of its constrained commit action.

Only checked candidates enter search. For each $a \in K_t^0$, DARP builds a depth-H recovery tree. With milestone, transition, and outcome estimators $\widehat{U}_q$, $\widehat{T}$, and $\widehat{p}$, where z ranges over possible tool outcomes, its value recursion is

$$\widehat{V}_0(s; q) = \widehat{U}_q(s), \tag{10}$$

$$\widehat{Q}_h(s, a; q) = -\eta c(a) + \sum_z \widehat{p}(z \mid s, a) \widehat{V}_{h-1}(\widehat{T}(s, a, z); q), \tag{11}$$

$$\widehat{V}_h(s; q) = \max_{a \in K^0(s)} \widehat{Q}_h(s, a; q). \tag{12}$$

Each search node carries its simulated policy and held capabilities; $K^0(s)$ abbreviates the candidates generated and filtered by Eq. (4) at that node. The recursion estimates the same first-action value as Eq. (6), using modeled transitions rather than observed returns. Known transitions and utility predicates yield the exact benchmark score. The automatic setting supplies schema-validated model estimates to the same recursion.

Near-maximal least-privilege selection. For a fixed tolerance $\epsilon \geq 0$, let $\widehat{Q}_{\max} = \max_{a \in K_t^0} \widehat{Q}_H(s_t, a; q)$. DARP retains the frontier

$$K_t^\epsilon = \{a \in K_t^0 : \widehat{Q}_H(s_t, a; q) \geq \widehat{Q}_{\max} - \epsilon\}. \tag{13}$$

The permission catalog maps each scope to the API methods it exposes using frozen Google Discovery documents. The coverage set contains direct method grants, scope grants, and the methods reachable through those scopes. We assign unit weight to every coverage element and no additional scope weight, so $W(S) = |S|$. The exposure introduced by a request and the direct footprint of an action are

$$E_{\mathrm{req}}(a \mid C_t) = W(\mathrm{Cov}(C_t \cup \mathrm{Req}(a)) \setminus \mathrm{Cov}(C_t)), \tag{14}$$

$$E_{\mathrm{cap}}(a) = W(\mathrm{Cap}(a)). \tag{15}$$

Thus E_{req} measures the additional coverage a request would make reachable if approved, while E_{cap} measures the direct capability footprint of the current action. A schema-aware repair distance compares operation, argument, and effect changes after masking fields named by the failed predicates:

$$D(a, a_t^-) = \tfrac{1}{3}(d_{\text{op}} + d_{\text{arg}} + d_{\text{eff}}). \tag{16}$$

Here d_{op} is a binary operation mismatch, d_{arg} is the fraction of unmasked argument fields that differ, and d_{eff} is the Jaccard distance between unmasked effect sets. Empty comparison sets have distance zero. All three components lie in $[0, 1]$; the primary evaluation uses equal weights fixed in advance. The lexicographic key is

$$r_t(a) = \big(E_{\text{req}}(a \mid C_t), E_{\text{cap}}(a), c(a), D(a, a_t^-), \text{id}(a)\big), \tag{17}$$

and the selected continuation is

$$a_t^* = \arg\min_{a \in K_t^\epsilon} r_t(a). \tag{18}$$

Recovery value determines membership in the frontier. Capability, cost, and repair distance order its members; the canonical identifier resolves any remaining tie.

An evidence read or authorization request may make little immediate task progress yet enable a later legal action. Bounded search values the resulting continuation, including any trusted approval outcome, so such a prerequisite can remain preferable to stopping on a recoverable task.

State-bound dispatch and guarantees. Selection creates a one-time certificate

$$\kappa_t = \big(n_t, F_a(a_t^*), F_s(s_t, C_t, P_t), \nu(P_t), F_\Omega(\Omega), b_t\big), \tag{19}$$

where F_a fingerprints the exact action, F_s covers the trusted state, held capabilities, policy contents, and relevant history, and F_Ω binds the registered tool interface, b_t records successful selection-time admissibility, and n_t is a unique identifier. Let $\mathcal{N}_{\text{used}}$ be the gateway's consumed nonce set. The agent proposes actions; the trusted gateway alone issues certificates and accesses the executor. A superscript κ_t denotes a value stored in the certificate. The gateway accepts exactly when

$$\begin{aligned} \mathcal{V}(\kappa_t, a, s_t, C_t, P_t, \Omega) = 1 \iff b_t = 1 \wedge n_t \notin \mathcal{N}_{\text{used}} \\ \wedge\, F_a(a) = F_a^{\kappa_t} \\ \wedge\, F_s(s_t, C_t, P_t) = F_s^{\kappa_t} \\ \wedge\, \nu(P_t) = \nu(P_t)^{\kappa_t} \\ \wedge\, F_\Omega(\Omega) = F_\Omega^{\kappa_t} \\ \wedge\, \text{Adm}(a, s_t, P_t) = 1. \end{aligned} \tag{20}$$

Successful verification atomically consumes n_t and dispatches the bound action. The observed return replaces predicted outcomes:

$$y_t = \text{Exec}(a_t^*), \qquad s_{t+1} = \text{Update}(s_t, y_t). \tag{21}$$

The ordinary agent replans from s_{t+1}; a later denial produces a new record, search, and certificate.

The guarantees use three conditions.

Assumption 1 (Checker soundness)**.** A successful Adm decision correctly interprets the deployed policy, semantic capabilities, and scope coverage.

Assumption 2 (Atomic certified dispatch)**.** Only the trusted gateway issues certificates and reaches the executor. Canonical fingerprints are collision resistant for the threat model. Issuance records the checked tuple; verification, nonce consumption, and dispatch are atomic with respect to the tuple current at dispatch.

Assumption 3 (Uniform value error)**.** For every $a \in K_t^0$,

$$\left|\widehat{Q}_H(s_t, a; q) - V_H(s_t, a; q)\right| \le \delta. \tag{22}$$

Proposition 1 (Certified dispatch safety)**.** Under Assumptions 1 and 2, each certified DARP action is legal in its dispatch state. A change to any bound field invalidates its certificate, which cannot be accepted twice. If every proposal is checked and every recovery step is certified, every action in a finite executed prefix is admissible under the state and policy current at its own dispatch.

Proposition 2 (Candidate-relative regret)**.** Under Assumption 3, Eq. (18) satisfies

$$\text{Reg}_H(a_t^*) \le \epsilon + 2\delta. \tag{23}$$

Proofs appear in Appendix A. The first proposition concerns the checker and dispatch gateway; the second concerns value-guided selection within K_t^0. Candidate coverage determines how much recovery value is available within that set.

4 EXPERIMENTS

We begin with a common authorization boundary and exact registered values to isolate the recovery choice after denial. We then vary the selection rule, search budget, value estimates, and candidate generator before checking that the selected action remains valid at dispatch.

Benchmark and protocol. We construct a typed benchmark for post-denial legal continuation from author-defined workflow templates, tool schemas, authorization conditions, and behavior-level evaluation labels. DARP-POSTDENIAL-480 represents the state immediately after a tool action is denied. Each task contains an authenticated objective, trusted state, held capabilities, policy, tool schemas, denied proposal, and failed checker predicates. Thirty workflow templates span Gmail, Drive, Calendar, and cross-application tasks. Two resource instances and eight denial conditions per template yield 480 tasks. Five templates form the pilot set, five form the development set, and 20 form the 320-task template-held-out test set, containing 280 recoverable and 40 unrecoverable cases. A separate 160-task compositional OOD split uses ten unseen workflow templates while retaining the tool vocabulary and permission semantics.

All non-ablation methods receive the same state, checker output, tool registry, three-step execution budget, certificate gateway, and sandbox executor. **Refuse** stops; **Direct Retry** generates one revised action; **Self-Revision** first audits the failed predicates; **Constraint Filter** selects the legal candidate with the highest cost-regularized bounded-horizon score ($\eta = 0.1$); and **Minimum Scope** prioritizes the least-exposure authorization request. We also implement **Progent-Fallback** (Shi et al., 2025) and checker-backed **ToolSafe TS-Flow** (Mou et al., 2026) as feedback-and-replanning baselines. They share the Qwen backbone and all execution components with DARP.

The development backbone is `qwen3.7-plus-2026-05-26`, with temperature 0, top-$p = 0.8$, a 2,048-token output limit, at most eight root candidates, and one format-repair attempt. We set $H = 3$, $\epsilon = 0.1$, and $\eta = 0.1$, then freeze the benchmark, prompts, checker, and controller before GPT and Claude transfer runs. Controlled comparisons use values computed from registered deterministic transitions and utility milestones; a separate experiment uses a frozen model scorer. Appendix D gives the complete artifact, visibility, baseline, and metric specifications.

Task Recovery Rate (TRR) measures completion among recoverable tasks, Correct Stop Rate (CSR) measures correct termination among unrecoverable tasks, and Safe Resolution Rate (SRR) combines both outcomes. Candidate Recall (CR) and First-Action Accuracy (FAA) measure whether a behavior-equivalent gold action is available and selected. Exposure counts newly reachable scopes and API methods along the executed path. Unauthorized Dispatch Rate (UDR) counts inadmissible executions, and Blocked Dispatch Rate (BDR) counts selected actions rejected by final revalidation. We use 50,000 paired bootstrap samples over the 20 held-out workflow templates for final pairwise intervals.

Recovery under a common authorization boundary. Table 1 compares recovery policies on the template-held-out split. The methods share an authorization boundary but make different choices after denial.

Method	TRR↑	CSR↑	SRR↑	CR↑	FAA↑	Exp.↓	Cost↓	BDR↓
Refuse	0.0	**100.0**	12.5	12.5	12.5	0.00	0.000	0.00
Direct Retry	84.3	95.0	85.6	73.4	72.8	7.97	0.676	1.25
Self-Revision	96.4	92.5	95.9	72.8	71.9	8.24	0.752	1.56
Constraint Filter	**100.0**	**100.0**	**100.0**	**100.0**	79.1	10.20	0.775	**0.00**
Minimum Scope	72.9	**100.0**	76.2	**100.0**	75.0	11.49	0.691	**0.00**
DARP	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**7.44**	0.775	**0.00**

Table 1: Results on 320 template-held-out tasks. Rates are percentages; exposure and cost are means. UDR is zero for every mediated method.

DARP resolves all 280 recoverable and 40 unrecoverable tasks. Constraint Filter attains the same SRR but raises mean exposure from 7.44 to 10.20; the near-maximal frontier and lexicographic selector reduce this exposure by 27.0%. Minimum Scope recovers 72.9% because the narrowest immediate request can block a useful multi-step path. The paired SRR gains over Direct Retry and Self-Revision are 14.38 points ($[10.00, 19.06]$) and 4.06 points ($[2.19, 5.94]$). Relative to Constraint

Split	Method	TRR↑	CSR↑	SRR↑	FAA↑	Exp.↓	Calls↓	Tokens↓
ID	DARP	**100.00**	**100.00**	**100.00**	**100.00**	**7.44**	**467**	**481.9k**
ID	Progent-Fallback	99.64	**100.00**	99.69	87.50	8.14	512	586.9k
ID	ToolSafe TS-Flow	99.64	**100.00**	99.69	87.50	8.01	505	588.1k
OOD	DARP	**100.00**	**100.00**	**100.00**	**100.00**	**8.51**	**233**	**256.6k**
OOD	Progent-Fallback	99.29	**100.00**	99.38	96.88	9.60	266	321.1k
OOD	ToolSafe TS-Flow	99.29	**100.00**	99.38	95.00	9.43	265	324.5k

Table 2: Comparison with published post-denial recovery mechanisms. ID has $N = 320$ and OOD has $N = 160$. Calls and tokens include online generation. UDR and API error rate are zero in all runs.

Filter, the paired exposure difference is -2.76 ($[-4.20, -1.38]$). Complete intervals appear in Appendix B.

The main comparison separates eventual completion from the quality of the first legal step. Table 2 asks whether that distinction persists against feedback-driven recovery policies on both workflow splits, using the same Qwen backbone, checker, tool schemas, gateway, and executor.

On ID, DARP improves FAA over Progent-Fallback and ToolSafe TS-Flow by 12.50 points; the paired 95% intervals are $[10.00, 14.69]$ and $[9.38, 15.63]$. The OOD gains are 3.13 points ($[0.63, 5.63]$) and 5.00 points ($[1.25, 9.38]$). Across both splits, DARP also makes 7.5–12.4% fewer API calls and uses 17.9–20.9% fewer tokens than these baselines. Appendix E reports the corresponding trade-off plot and Self-Revision transfer control.

What produces the gain? The baseline comparison motivates three complementary tests of the design: authorization mediation, preference ordering within the legal set, and execution binding. Table 3 compares these variants.

Variant	TRR	SRR	Exp.	UDR	AAR
DARP	100.0	100.0	7.44	0.0	0.0
Task utility only	89.6	90.9	10.63	0.0	–
No auth. gate	71.4	25.0	0.00	75.0	–
Without binding	100.0	100.0	7.44	0.0	100.0

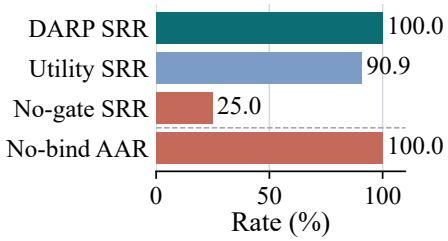
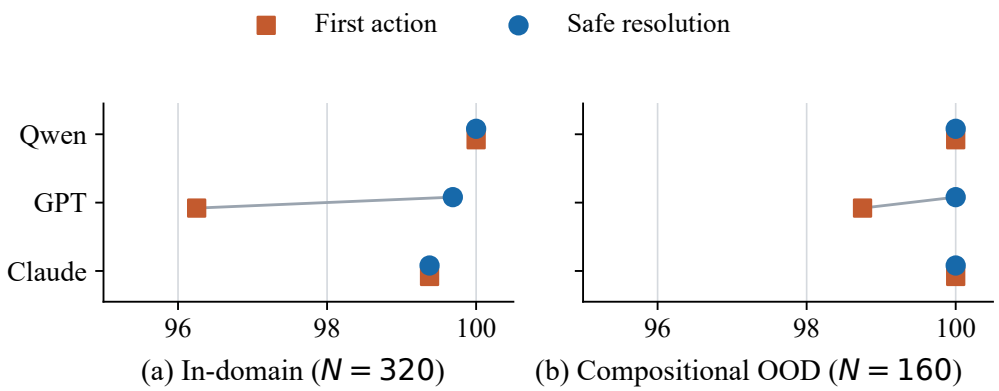


Table 3: Component ablations. "No auth. gate" removes legal projection and final authorization mediation. AAR is attack acceptance rate; all rates are percentages.

Task-utility-only selection sets $\eta = 0$ and takes an exact maximizer of task return among legal candidates, with only a canonical tie break. It therefore removes both the near-maximal frontier and the capability, execution-cost, and repair ordering used by DARP; TRR falls by 10.4 points and exposure rises by 3.18. Constraint Filter instead retains the standard cost-regularized bounded-horizon score but applies a hard argmax, explaining why it preserves recovery while requesting more capability. Removing the authorization gate disables both projection through Adm and certificate-gateway enforcement, producing 75.0% inadmissible dispatches and 25.0% SRR. Removing state binding preserves benign outcomes but accepts every tested post-selection perturbation. The full gateway rejects all 2,100 argument, evidence, permission, version, and replay perturbations. Appendix F reports results by attack class.

The selector can only compare continuations reached by the search. SRR rises from 31.25% at $H = 1$ to 75.00% at $H = 2$ and 100% at $H = 3$. A candidate budget of four is the smallest tested value reaching 100% SRR; mean controller latency is 1.355 ms. The candidate pool contains 3.59 actions per ID decision and 3.65 per OOD decision on average, with 100% candidate recall on both splits. Detailed depth, budget, latency, and pool statistics are in Appendix G.



(a) In-domain ($N = 320$) (b) Compositional OOD ($N = 160$)

Figure 2: First-action accuracy and safe resolution for candidate pools from three models on the in-domain and compositional OOD splits. Rates are percentages; the controller and legal checker are fixed.

Approximate values and transfer. The preceding comparisons use values computed from registered transitions. Replacing them with a frozen Qwen scorer tests the value estimate while leaving candidate construction, legal projection, and dispatch unchanged.

The scorer reaches 0.164 MAE and 0.835 Spearman correlation over 1,053 actions. It obtains 90.94% SRR while every unresolved recoverable case stops legally and UDR remains zero. In the separate bounded-noise sweep, outcomes remain unchanged through $\delta = 0.05$, and all 17,292 recovery decisions satisfy the $\epsilon + 2\delta$ candidate-relative regret bound. Appendix H gives the full sweep.

Source	TRR	SRR	FAA	UDR
Exact	100.00	100.00	100.00	0.00
Frozen Qwen	89.64	90.94	86.25	0.00

Table 4: Exact and automatically estimated action values.

Value estimates change how a fixed legal set is ordered; a different candidate model changes the set itself. Table 5 evaluates the frozen controller on candidate sets generated by Qwen, GPT, and Claude.

	In-domain ($N = 320$)			Compositional OOD ($N = 160$)		
Candidate model	SRR	FAA	Exp.	SRR	FAA	Exp.
Qwen3.7-Plus	100.00	100.00	7.44	100.00	100.00	8.51
GPT-5.6-sol	99.69	96.25	7.44	100.00	98.75	8.51
Claude Opus 5	99.38	99.38	7.44	100.00	100.00	8.51

Table 5: DARP transfer across candidate models and workflow splits. CSR is 100%, while UDR, BDR, and API error rate are zero in all six runs.

All three candidate models reach at least 99.38% ID SRR and 100% OOD SRR. GPT misses the behavior-equivalent first action in 12 ID and two OOD tasks, yet alternative legal paths recover all but one ID task. Replaying each frozen model-specific pool through Constraint Filter preserves SRR up to rounding. Relative to this control, DARP lowers exposure by 20.4–27.3% on the same candidate sets. Full frozen-candidate results appear in Appendix E.

Figure 2 places first-action accuracy beside safe resolution for each candidate model. The gaps in the GPT runs show that a different legal first step can still lead to task completion; the table retains the corresponding capability-exposure values.

Validity and execution audit. Selection alone does not establish what the executor receives. Table 6 follows the chosen actions through information-flow, rolling-execution, and certificate checks, alongside a blinded human assessment of their utility.

Audit	Units	Result
Public views and API requests	640 / 4,268	0 prohibited fields exposed
Hidden-label perturbation	640 / 480	public and candidate hashes stable
Sandbox trajectories and transitions	32 / 63	all passed and matched
Fresh / replayed / stale certificates	43 / 43 / 43	all decisions correct
Human recoverability	80	$\kappa = 1.00$
Admissible / useful sets	80	100% / 100% exact agreement
DARP useful / minimal / best	160	100% / 100% / 100%
Minimum-set exact agreement	80	75.0%

Table 6: Validity, rolling execution, and blinded human audit. Human set agreement reports admissible and useful sets.

No prohibited evaluation field appears in 4,268 recorded model requests. Perturbing hidden labels leaves the public-view, prompt, cache, and candidate hashes unchanged, so the tested candidate construction does not depend on those evaluator-only labels. The execution audit then follows selected actions through observed returns. Its 32 tasks include 16 from each split, eight from each application domain, and four from each denial condition. Twenty-eight require recovery and four require a correct stop; all 32 trajectories resolve safely, and all 63 reconstructed transitions match the execution log. Fresh, replayed, and stale certificates each receive the correct decision in all 43 tests of their respective class, exercising the state and one-time bindings at the point of dispatch.

The execution and human audits examine different parts of the same recovery decision. Certificate and transition checks establish which action reaches the tool and how the state changes, but they do not distinguish among the legal alternatives left by the checker. The candidate pools contain more than two admissible actions per recovery decision on average in both workflow splits. The blinded audit evaluates this choice through independent judgments of usefulness and minimum capability.

Its 80 cases include 40 from each workflow split, balanced across denial configurations. Annotators saw the objective, current state, checker feedback, and candidate actions, but not method names, the action chosen by DARP, gold actions, recoverability labels, or scores. Candidate order was independently randomized for each case. Annotators first marked the admissible, useful, and minimum-capability sets, then chose one action by task progress, capability expansion, execution cost, and deviation from the denied proposal. Across these cases, both annotators agree on recoverability and admissibility and select the DARP action as useful, minimum-capability, and best every time. Their full minimum-capability option sets agree in 60 cases; in the remaining 20, one set is a strict subset of the other, while both annotators still choose the same best action. The difference concerns which alternatives also qualify as minimal, rather than the selected continuation. Appendix C details the protocol.

5 Conclusion and Limitations

Post-denial legal continuation treats a rejected tool call as a new decision point while the user's task remains unfinished. DARP turns structured denial feedback into typed alternatives, compares their recoverable value within the legal set, and limits capability use among near-maximal choices. State-bound certification carries the selected action through dispatch before the agent replans from the observed return. These steps give authorization and recovery distinct roles: the checker determines which action may execute, while the selector determines which permitted action best advances the task. The selection guarantee is relative to the generated candidates; execution safety relies on sound checking and atomic dispatch, and the value bound also requires bounded estimation error. Extending this decision rule to stochastic services and changing permission models remains an open direction.

AI use statement

Generative AI tools were used to aid and polish the writing and to support literature retrieval and discovery. The authors reviewed and revised all AI-assisted text, verified cited sources against the original publications, and take responsibility for the final content of this work.

REPRODUCIBILITY

We retain the scripts used to build the fixed in-domain and OOD benchmarks, the resulting task artifacts, experiment configurations, and run-level records underlying the reported results. The evaluation entry point and settings are in `main.py` and `config.py`. Appendix D describes benchmark construction and the common evaluation protocol, while Appendix C details the human audit. Repeating model-based runs requires access to the corresponding APIs.

REFERENCES

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. Agentpoison: Red-teaming llm agents via poisoning memory or knowledge bases. *Advances in Neural Information Processing Systems*, 37:130185–130213, 2024.

Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. *Advances in neural information processing systems*, 37:82895–82920, 2024.

Edoardo Debenedetti, Ilia Shumailov, Tianqi Fan, Jamie Hayes, Nicholas Carlini, Daniel Fabian, Christoph Kern, Chongyang Shi, Andreas Terzis, and Florian Tramèr. Defeating prompt injections by design. *arXiv preprint arXiv:2503.18813*, 2025.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Nan Duan, Weizhu Chen, et al. Critic: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations*, volume 2024, pp. 57734–57811, 2024.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM workshop on artificial intelligence and security*, pp. 79–90, 2023.

Feiran Jia, Tong Wu, Xin Qin, and Anna Squicciarini. The task shield: Enforcing task alignment to defend against indirect prompt injection in llm agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 29680–29697, 2025.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. When can llms actually correct their own mistakes? a critical survey of self-correction of llms. *Transactions of the Association for Computational Linguistics*, 12:1417–1440, 2024.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. Api-bank: A comprehensive benchmark for tool-augmented llms. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 3102–3116, 2023.

Junda Lin, Zhaomeng Zhou, Zhi Zheng, Shuochen Liu, Tong Xu, Yong Chen, and Enhong Chen. Vigil: Defending llm agents against tool-stream injection via verify-before-commit. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9764–9785, 2026.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pp. 52989–53046, 2024.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, et al. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183, 2025.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Yutao Mou, Zhangchi Xue, Lijun Li, Peiyang Liu, Shikun Zhang, Wei Ye, and Jing Shao. Toolsafe: Enhancing tool invocation safety of llm-based agents via proactive step-level guardrail and feedback. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 37125–37153, 2026.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37: 126544–126565, 2024.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations*, volume 2024, pp. 9695–9717, 2024.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris Maddison, and Tatsunori Hashimoto. Identifying the risks of lm agents with an lm-emulated sandbox. In *International Conference on Learning Representations*, volume 2024, pp. 27031–27098, 2024.

Jerome H Saltzer and Michael D Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Fred B Schneider. Enforceable security policies. *ACM Transactions on Information and System Security (TISSEC)*, 3(1):30–50, 2000.

Tianneng Shi, Jingxuan He, Zhun Wang, Hongwei Li, Linyu Wu, Wenbo Guo, and Dawn Song. Progent: Securing ai agents with privilege control. *arXiv preprint arXiv:2504.11703*, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. Adaplanner: Adaptive planning from feedback with language models. *Advances in neural information processing systems*, 36:58202–58245, 2023.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv preprint arXiv:2306.05301*, 2023.

Kim Peter Wabersich and Melanie N Zeilinger. A predictive safety filter for learning-based control of constrained nonlinear dynamical systems. *Automatica*, 129:109597, 2021.

Yuhao Wu, Franziska Roesner, Tadayoshi Kohno, Ning Zhang, and Umar Iqbal. Isolategpt: An execution isolation architecture for llm-based agentic systems. *arXiv preprint arXiv:2403.04960*, 2024.

Qiantong Xu, Fenglu Hong, Bo Li, Changran Hu, Zhengyu Chen, and Jian Zhang. On the tool manipulation capability of open-source large language models. *arXiv preprint arXiv:2305.16504*, 2023.

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.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Junjie Ye, Sixian Li, Guanyu Li, Caishuang Huang, Songyang Gao, Yilong Wu, Qi Zhang, Tao Gui, and Xuan-Jing Huang. Toolsword: Unveiling safety issues of large language models in tool learning across three stages. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2181–2211, 2024.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10471–10506, 2024.

Hanrong Zhang, Jingyuan Huang, Kai Mei, Yifei Yao, Zhenting Wang, Chenlu Zhan, Hongwei Wang, and Yongfeng Zhang. Agent security bench (asb): Formalizing and benchmarking attacks and defenses in llm-based agents. In *International Conference on Learning Representations*, volume 2025, pp. 35331–35366, 2025.

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.

Dongsheng Zhu, Xuchen Ma, Yucheng Shen, Xiang Li, Yukun Zhao, Shuaiqiang Wang, Lingyong Yan, and Dawei Yin. When tools fail: Benchmarking dynamic replanning and anomaly recovery in llm agents. *arXiv preprint arXiv:2606.05806*, 2026.

Jinhao Zhu, Kevin Tseng, Gil Vernik, Xiao Huang, Shishir G Patil, Vivian Fang, and Raluca Ada Popa. Miniscope: A least privilege framework for authorizing tool calling agents. *arXiv preprint arXiv:2512.11147*, 2025.

A PROOFS

Proof of Proposition 1. Equation (18) selects from $K_t^\epsilon \subseteq K_t^0$. By Eq. (4), its members pass the policy, held-capability, and permitted-request checks. The certificate records this selection-time result in b_t. At dispatch, Eq. (20) rechecks Adm and compares the action, trusted state, held capabilities, policy version, and tool interface with their certified values. Under Assumption 2, these comparisons and dispatch observe the same tuple; Assumption 1 supplies the policy interpretation. Thus the selected action remains in the legal set of its dispatch state. Collision resistance detects changes to bound fields, and atomic insertion of n_t into $\mathcal{N}_{\mathrm{used}}$ prevents a second acceptance. Finally, induction on the executed prefix applies the current-step argument to each recovery action. An original proposal is dispatched only after its own current-state check. Observed returns create the next state, where the check is repeated. Hence every finite executed prefix is admissible step by step. □

Proof of Proposition 2. Let a_t^{opt} maximize the true value in K_t^0. Since a_t^* belongs to K_t^ϵ and $\widehat{Q}_{\max} \geq \widehat{Q}_H(s_t, a_t^{\mathrm{opt}}; q)$,

$$\widehat{Q}_H(s_t, a_t^*; q) \geq \widehat{Q}_H(s_t, a_t^{\mathrm{opt}}; q) - \epsilon. \tag{24}$$

Applying Assumption 3 once to each action gives

$$V_H(s_t, a_t^*; q) \geq \widehat{Q}_H(s_t, a_t^*; q) - \delta \tag{25}$$

$$\geq \widehat{Q}_H(s_t, a_t^{\mathrm{opt}}; q) - \epsilon - \delta \tag{26}$$

$$\geq V_H(s_t, a_t^{\mathrm{opt}}; q) - \epsilon - 2\delta. \tag{27}$$

Rearranging and applying Eq. (7) yields $\mathrm{Reg}_H(a_t^*) \leq \epsilon + 2\delta$. □

B PAIRED STATISTICAL AUDIT

We resample the 20 held-out ID workflow templates with replacement and retain all task instances within each sampled template. Each replicate therefore preserves pairing between DARP and the comparison method. Table 7 reports the difference between DARP and each baseline over 50,000 replicates. Rate differences are percentage points. Negative exposure differences indicate that DARP requests less semantic capability.

Baseline	ΔTRR [95% CI]	ΔSRR [95% CI]	ΔExp. [95% CI]
Refuse	100.00 [100.00, 100.00]	87.50 [87.50, 87.50]	7.44 [5.14, 9.90]
Direct Retry	15.71 [10.36, 21.43]	14.38 [10.00, 19.06]	−0.53 [−1.31, 0.00]
Self-Revision	3.57 [1.79, 5.36]	4.06 [2.19, 5.94]	−0.79 [−1.62, −0.14]
Constraint Filter	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]	−2.76 [−4.20, −1.38]
Minimum Scope	27.14 [24.29, 28.57]	23.75 [21.25, 25.00]	−4.05 [−5.31, −3.01]

Table 7: Paired template-cluster bootstrap intervals on the 320-task ID test split. Differences are DARP minus baseline.

C HUMAN AUDIT PROTOCOL

The audit samples 40 ID and 40 OOD cases, balanced across denial configurations. Each case presents the authenticated objective, trusted state, current policy and capabilities, checker failures, relevant tool contracts, and the candidate actions available at the first recovery decision. Candidate order is independently randomized for every case. The casebook omits method names, the action selected by DARP, benchmark recoverability labels, gold actions, scores, and exposure values.

Two annotators label each case independently. They first determine whether a legal continuation remains, then identify the admissible, useful, and minimum-capability option sets. Finally, they choose one current action by task progress, capability expansion, execution cost, and deviation from the denied proposal, in that order. The reported agreement and selection rates are computed directly from the two annotation files without adjudicating disagreements. This protocol evaluates whether the checker labels and the continuation chosen by DARP agree with independent judgments under the same stated recovery ordering.

D Benchmark and Evaluation Details

Each benchmark item stores behavior-level gold first actions, recoverability labels, and measured task outcomes as evaluator-only fields. They are excluded from candidate-generation requests, which also omit deterministic transitions and utility predicates. The automatic value scorer receives the registered transitions and utility specification needed by the bounded-horizon recursion, but not gold actions, recoverability labels, or measured outcomes.

The benchmark registers 32 Google Workspace methods. Method–scope relations come from frozen Gmail, Drive, and Calendar Discovery documents. The typed sandbox validates arguments and applies deterministic transitions without modifying a live account. Build-time checks cover artifact hashes, split isolation, denial labels, legal-set non-emptiness, and inclusion of safe stop. The OOD split introduces unseen objective, operation, and evidence compositions across ten held-out templates.

Shared-candidate methods use the same frozen candidate artifact. Registered control actions are instantiated deterministically, while the candidate model generates task-specific repairs, reads, scope reductions, and staging actions. Direct Retry and Self-Revision use method-specific prompts. Every selected action passes through the same parser, checker, certificate gateway, and executor. Progent-Fallback and ToolSafe TS-Flow generate one continuation per feedback round for at most three rounds. API calls and tokens include online candidate generation.

Cost is the mean cumulative executed-path value obtained by summing Eq. (5). Exposure uses the frozen Discovery-derived method–scope map, unit weight for each coverage element, and no additional base weight for scopes. UDR counts inadmissible actions reaching execution; BDR counts selected actions rejected by final revalidation. Candidate Recall and FAA use behavior-equivalent, rather than string-identical, first actions.

E Supplementary Recovery Controls

Model	Split	SRR	Exposure
Qwen	ID	100.00	10.20
Qwen	OOD	100.00	10.86
GPT	ID	99.69	9.89
GPT	OOD	100.00	10.69
Claude	ID	99.38	10.23
Claude	OOD	100.00	11.04

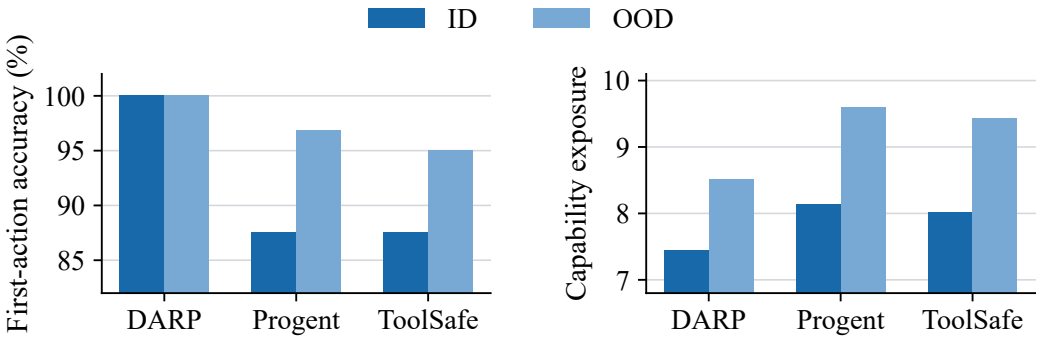
Table 8: Constraint Filter on frozen candidate artifacts. CSR is 100%, and UDR and BDR are zero in all six runs.

Split	N	SRR	Exposure
ID	320	95.9	8.24
OOD	160	95.0	9.50

Table 9: Self-Revision across workflow splits.

Self-Revision reaches 95.9% SRR on ID and 95.0% on OOD, with exposure 8.24 and 9.50. The corresponding DARP results are 100% SRR with exposure 7.44 and 8.51.

Constraint Filter preserves the SRR of each frozen candidate pool up to rounding. Relative to this control, DARP lowers exposure by 20.4–27.3% with generation fixed, isolating the near-maximal frontier and lexicographic selector.



(a) First-action accuracy (b) Capability exposure

Figure 3: Quality and capability use for published recovery baselines on the ID and compositional OOD splits.

F Dispatch Integrity Controls

Each perturbation is applied to 420 certificates, giving 2,100 modified dispatches. The full gateway rejects every mutation while accepting every unmodified control. The no-binding variant retains selection-time checking but omits the action, state, policy-version, tool-version, and replay bindings in Eq. (20); it accepts every perturbation.

Perturbation	Full	No binding
Argument mutation	0.0	100.0
Evidence invalidation	0.0	100.0
Permission revocation	0.0	100.0
Policy/tool version	0.0	100.0
Replay	0.0	100.0

Table 10: Attack acceptance rate over 420 certificates per perturbation.

G Search and Candidate Controls

H	TRR	SRR	Exposure
1	21.43	31.25	0.000
2	71.43	75.00	2.481
3	100.00	100.00	7.444

Table 11: Effect of planning horizon on the 320-task test set.

K	SRR	Exposure	Mean ms	P95 ms
1	88.13	9.47	1.245	2.645
2	99.38	7.58	1.314	2.823
4	100.00	7.44	1.355	2.969
8	100.00	7.44	1.378	3.014

Table 12: Candidate budget and controller latency on the ID test set.

At $H = 1$, legal prerequisites with no immediate task reward lose to stopping. Increasing the horizon to two recovers 71.43% of recoverable tasks, and $H = 3$ covers every registered recovery template.

Across 3,840 API-free timing episodes, no controller error or blocked dispatch occurs. Candidate pools contain 2–6 actions per recovery decision. Their mean sizes are 3.59 on ID and 3.65 on OOD, with 2.59 and 2.65 legal candidates on average. Candidate recall is 100% on both splits. The model contributes 1.11 candidates per ID state and 1.10 per OOD state, alongside registered controls and safe stop. In the human-audit sample, 3.53 displayed actions contain 2.53 admissible and 1.65 useful candidates on average.

H Value Robustness

Every exact action value is perturbed by independent bounded noise $|\xi(a)| \leq \delta$. Results are unchanged through $\delta = 0.05$ and decline gradually thereafter. At $\delta = 0.20$, mean SRR is 97.91%

δ	TRR (%)	SRR (%)	Exposure	Mean regret	Max regret	Violations
0.00	100.00	100.00	7.444	0	0	0/420
0.02	100.00 (0.00)	100.00 (0.00)	7.444 (0.000)	0	0	0/4,200
0.05	100.00 (0.00)	100.00 (0.00)	7.444 (0.000)	0	0	0/4,200
0.10	99.82 (0.18)	99.84 (0.16)	8.448 (0.269)	2.97×10^{-5}	0.025	0/4,205
0.20	97.61 (1.38)	97.91 (1.21)	9.594 (0.339)	3.91×10^{-4}	0.025	0/4,267

Table 13: Robustness to bounded value perturbations. Parentheses report population standard deviations over ten seeds. A violation occurs when exact candidate-relative regret exceeds $\epsilon + 2\delta$.

with standard deviation 1.21 and exposure is 9.594 ± 0.339. Across 17,292 recovery decisions, maximum exact regret is 0.025 and no decision violates the $\epsilon + 2\delta$ bound. Separate sweeps over ϵ, η, and three repair-distance profiles preserve the primary trajectory; 194 of 420 recovery decisions contain multiple near-maximal actions.

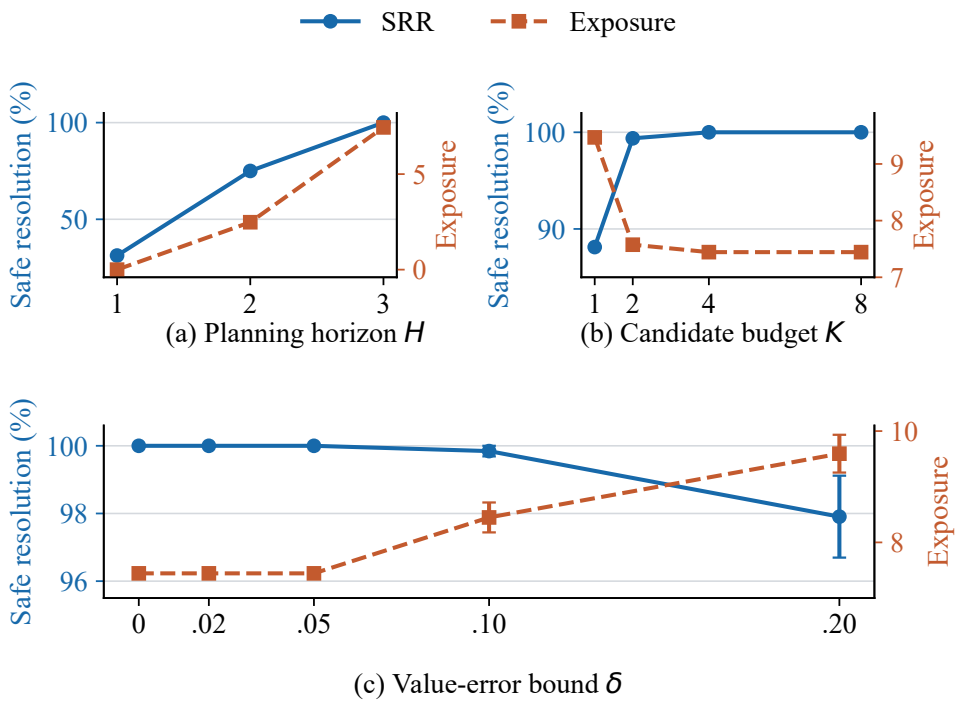


Figure 4: Sensitivity to planning horizon, candidate budget, and bounded value error. Lines report safe resolution and mean capability exposure; error bars show population standard deviations over ten seeds.