SKIP THE MODEL, NOT THE CHECKS: CERTIFYING MODEL-FREE ACTIONS OF LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

To save inference cost, LLM agents increasingly act without the model: a lightweight proposer predicts the next tool and fills its arguments with values reused from earlier calls. Such actions are cheap but barely checked. We propose certify–repair–ask (CRA), a training-free gate that executes a proposed action only if its preconditions are certified in three-valued logic over observation-grounded predicates, otherwise executes the best certified alternative among the cached values, and asks the model only when none exists. We restore ScienceWorld and ALFWorld to 2339 decision points of an AutoTool agent, execute each gate's action there, and branch episodes from the same states. First, certification cuts refusals: with an expert catalog, CRA lowers the refusal rate of model-free actions from 80.2% and 93.7% to 28.5% and 26.6%, where a reject gate almost never acts. Second, executable is not the same as right: catalogued preconditions still hold after an action has taken effect, so repair replays the agent's recent past (where the model acted, 35.4% of CRA's accepted ScienceWorld actions re-execute a state change already made, against 3.9% of the model's, and every accepted ALFWorld navigation repair returns to the previous location), and branched from the same states they cost ALFWorld agents extra steps and calls without detectable success gains. Third, in closed loop no gate detectably improves success or model calls over unguarded reuse or native AutoTool, and in ALFWorld repair lowers success relative to rejection, clearly with Claude, less so with Gemini; asking instead of repeating an accepted action restores it. Our results suggest that gates on model-free actions need a progress-sensitive signal beyond executability: a refused action costs a step, an executable wrong one can cost the task.

1 INTRODUCTION

Large language model (LLM) agents solve multi-step tasks by emitting structured tool calls (Yao et al., 2023; Schick et al., 2023; Qin et al., 2024), and in the standard loop every step costs a model invocation. Because agent trajectories are long and repetitive, a growing family of systems cuts this cost by acting *without* the model whenever they can: they cache responses (Bang, 2023; Gaspar Schroeder et al., 2026), plan tool calls ahead of observations (Xu et al., 2023; Kim et al., 2024), route to cheaper models (Chen et al., 2023), or, as in AutoTool (Jia & Li, 2026), predict the next tool from usage statistics and fill its arguments with values that earlier calls produced. We call an action that is executed without querying the LLM a *model-free action*.

Model-free actions are cheap, but they are barely checked. AutoTool's reuse path verifies that every argument can be filled, not that the filled values are valid together or still valid now: a model-generated call is conditioned on the full context, whereas a model-free call is assembled from values that were valid when they were observed. Values are *mismatched*: at every one of 563 multi-argument decision points in our ScienceWorld (Wang et al., 2022) logs, slot-wise reuse put the same entity into several slots, as in `move(thermometer, thermometer)`. Values also go *stale*: a first version of our own checker verified only that a navigation target was a known room, and all 553 cached navigation actions it passed named the room the agent was already in. We ask:

Can an agent skip the model without skipping the checks?

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.

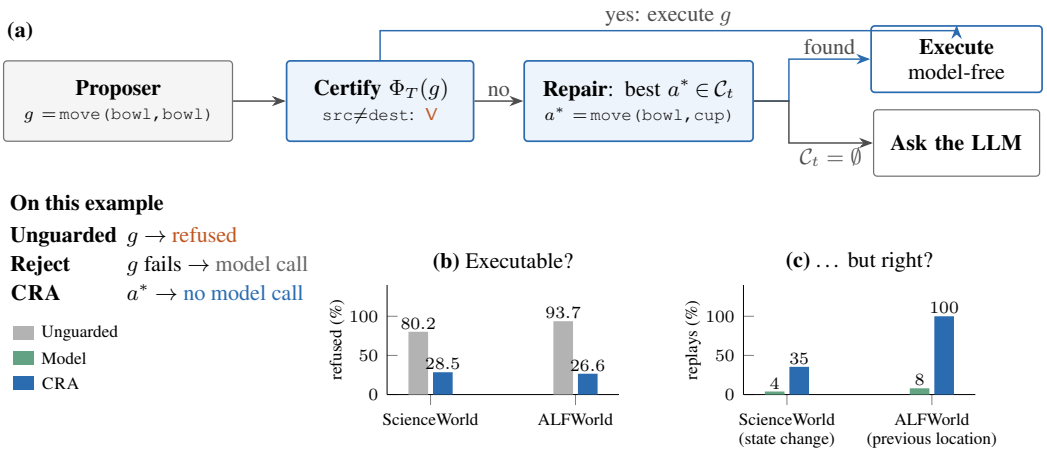


Figure 1: **Certify–repair–ask and what it achieves.** (a) CRA executes the proposer's greedy tuple g only if its preconditions are certified, otherwise the best certified cached tuple a^*, and otherwise asks the LLM; here both slots of `move` rank `bowl` first. (b) Certification with the expert catalog cuts refusals (refusal rate of model-free actions at the common decision points; Section 5.2). (c) But repair replays the recent past: accepted actions that re-execute a state change already made, at ScienceWorld points where the model acted, and accepted navigations back to the previous location (ALFWorld closed loop, Gemini), for the model and for CRA (Section 5.3).

The obvious safeguard, rejecting a proposal that fails a check and asking the model instead, gives up the saving: at our ScienceWorld decision points a reject gate never acts model-free, because every proposal fails, yet the cache contains a certified alternative at 36.2% of all 563. We propose *certify–repair–ask* (CRA; Figure 1a), which separates whether the *proposed* tuple is admissible from whether *any* cached tuple is. CRA (i) *certifies* a proposed tuple by evaluating its preconditions in strong Kleene three-valued logic (Kleene, 1952) over predicates grounded in the observation history, so missing evidence yields *Unknown* instead of a false pass; (ii) if certification fails, *repairs* the proposal by searching the cached candidates for the highest-ranked certified tuple; and (iii) otherwise *asks* the model. CRA needs no training, treats the proposer and the LLM as black boxes, and never changes an action that a reject gate would execute. Its predicates can come from an expert, from execution logs (Ammons et al., 2002; Ernst et al., 2007), or from the LLM itself, which writes them *once*, offline, from the tool documentation.

Measuring a gate is itself subtle: replayed logs cannot tell whether a changed action would execute, and closed-loop runs confound the gate with the stochastic policy it alters. We therefore restore ScienceWorld and ALFWorld (Shridhar et al., 2021) to 2339 decision points of an AutoTool agent and execute each gate's action from the same state (*counterfactual execution*); we branch whole episodes from single states (*same-state branches*); and we run the gated agent in closed loop with two LLMs. Together these answer three questions: do certificates make model-free actions executable (**RQ1**, Section 5.2), is an executable action the right one (**RQ2**, Section 5.3), and what does gating do to whole episodes (**RQ3**, Section 5.4)? Our results highlight three findings.

Certificates cut refusals. With the expert catalog, certification cuts the refusal rate of multi-argument model-free actions from 80.2% and 93.7% to 28.5% and 26.6% in ScienceWorld and ALFWorld (Figure 1b), where a reject gate rarely acts; uncertified repairs fare far worse.

Executable is not the same as right. Catalogued preconditions still hold after an action has taken effect, so recency-ranked repair replays the agent's recent past: at the ScienceWorld points where the model acted, 35.4% of the accepted actions of CRA re-execute a state change already made (3.9% of the model's do), and every accepted ALFWorld navigation repair returns to the previous location (Figure 1c). Branched from the same states, they do not pay off: a repaired ALFWorld navigation saves a model call but costs about two steps and more calls, and in ScienceWorld the model's own action succeeds numerically more often (56.2% vs. 49.0% of runs, CI [0.0, 17.8]).

Fewer refusals do not make better episodes. In closed loop, gating removes most refused model-free actions but improves neither success nor model calls detectably over unguarded reuse or native

AutoTool. In ALFWorld, where paired episodes stay coupled until gate decisions differ, CRA lowers success relative to rejection, most clearly with Claude; in ScienceWorld they decouple earlier through the nondeterministic model, hence our same-state branches. Asking instead of repeating an accepted action restores the success lost to repair (Claude: 9.7 points, p=0.002), without beating rejection.

We contribute (1) protocols that separate inadmissible from executable but wrong model-free actions (Sections 2 and 4); (2) CRA, which provably acts model-free wherever rejection does (Sections 2 and 3); and (3) evidence, with a mechanism and an intervention, that executability is not enough.

2 Model-free actions and how to gate them

Setting. An agent interacts with an environment for up to H steps. At step t it either queries the LLM, which returns an action, or acts model-free. A model-free action comes from a *proposer* that predicts a tool T with m argument slots and, for each slot k, a candidate list V_k of values taken from earlier successful calls and their outputs within a window of W records. Each candidate keeps its provenance (source call, field, entity type) and a recency score r_k (higher is more recent); lists are truncated to K entries. The greedy proposal g_t fills every slot with its top-ranked candidate, independently of the other slots. We call this our *shared filler*; it ranks by recency, whereas AutoTool's native filler follows a learned parameter-dependency graph (Jia & Li, 2026), and Section 5.4 runs both. We leave the proposer, the prompts, and the termination rules unchanged and study only the *gate* that decides what, if anything, to execute without the model. A *decision point* is a step at which the proposer can fill every slot of a tool with $m \geq 2$.

Certificates. Each tool T carries a check set $\Phi_T = \{\phi_1, \ldots, \phi_n\}$: record-validity and schema-type checks plus domain predicates such as *the source is present*, *the destination is a container*, or *source and destination differ*. Every ϕ_j maps a tuple $a \in V_1 \times \cdots \times V_m$ and the evidence E_t (the observation history up to t) to Satisfied (S), Violated (V), or Unknown (U). Checks are combined by the strong Kleene conjunction: $\Phi_T(a) = \mathsf{S}$ if every check is S, V if any is V, and U otherwise. A tuple is *certified* if $\Phi_T(a) = \mathsf{S}$; the certified set is $\mathcal{C}_t = \{a : \Phi_T(a) = \mathsf{S}\}$. Unknown is never read as Satisfied: a check that lacks evidence blocks certification rather than passing it. A tool without domain predicates is certified by its record checks alone (Section 5.4).

Gates. Writing $\perp$ for asking the LLM, we compare three gates:

$$G_{\mathrm{U}}(t) = g_t, \qquad G_{\mathrm{R}}(t) = \begin{cases} g_t & \Phi_T(g_t) = \mathsf{S} \\ \perp & \text{otherwise,} \end{cases} \tag{1}$$

$$G_{\mathrm{CRA}}(t) = \begin{cases} \arg\max_{a \in \mathcal{C}_t} s(a), \;\; s(a) = \sum_k r_k(a_k) & \mathcal{C}_t \neq \emptyset \\ \perp & \text{otherwise.} \end{cases} \tag{2}$$

The unguarded gate G_{U} is the behavior of a cache without checks, the reject gate G_{R} is the natural safeguard, and CRA adds repair, breaking ties in rank order so that a certified g_t is kept. We measure a gate by its *skip rate* (the fraction of decision points at which it acts model-free), its *refusal rate* (the fraction of its model-free actions that the environment refuses), and, at states where the model itself acted, how many of its accepted model-free actions differ from the model's action. A model-free action can fail in two ways: it can be *inadmissible*, so that the environment refuses it and a step is wasted, or it can execute and still differ from what the model would have done, which no check of preconditions can detect. Whether the skips add up to a net saving is a closed-loop question.

Guarantees. Two properties hold by construction, assuming that the repair search finishes within its budget, which it always did (proofs in Appendix A).

Proposition 1 (Dominance over rejection)**.** *If $G_{\mathrm{R}}(t) \neq \perp$ then $G_{\mathrm{CRA}}(t) = G_{\mathrm{R}}(t)$. Moreover $G_{\mathrm{CRA}}(t) = \perp$ if and only if $\mathcal{C}_t = \emptyset$. Hence* CRA *skips at least as often as rejection, and the additional skips are exactly the decision points with $g_t \notin \mathcal{C}_t \neq \emptyset$.*

Second, if every check is sound, i.e., $\phi_j(a) = \mathsf{S}$ implies its precondition, every tuple that G_{R} or G_{CRA} executes satisfies all catalogued preconditions, so a refusal stems from a precondition the catalog lacks; reading U as S loses this (Proposition 2). These are structural properties, not new

verification theory: for execution decisions, strong Kleene certification equals a conservative two-valued gate, and its third value separates missing from contradicting evidence (Section 5.2).

3 CERTIFY–REPAIR–ASK

CRA needs predicates whose Satisfied verdicts can be trusted in text environments, where the state is known only through observations, and a search that finds a certified tuple quickly.

3.1 GROUNDING PREDICATES IN OBSERVATIONS

A feedback sentence, a description variant, or an animal can pass a string test, so predicates must be grounded in what the agent has observed. We maintain four kinds of evidence from the observation history, compare canonical names (never substrings), and update evidence only from accepted actions: *identity*, *presence* (visible or carried, including qualified referents such as *thermometer in inventory*), *role* (a destination is a container only if an observation states containment, support, or a door), and *location*. A plain name absent from the visible and carried sets is Violated; everything else undecided is Unknown, including liquid state, so CRA with the expert catalog always asks before pouring and dunking. Appendix B gives the full rules.

3.2 THREE SOURCES OF CERTIFICATES

All three catalogs share the candidates, the grounded tracker, and name resolution; they differ in which predicates apply to which tool and argument (Appendix B). **Expert.** A hand-written catalog of 19 composite checks over 16 tools (before the ALFWorld navigation check below) for ScienceWorld, ALFWorld, and ToolQuery-Academic (Ma et al., 2024), with relational predicates such as *distinct*, *container target*, *object at receptacle*, and *destination differs from the current location*; we added the last after inspecting refusals in our development logs (Section 5.1) and, as a further check, its ALFWorld counterpart before the ALFWorld closed loop, freezing the catalogs before each closed-loop run. **Mined.** Following specification mining (Ammons et al., 2002; Ernst et al., 2007), we instantiate a template library (*visible*, *in inventory*, *reachable*, *container*, *known location*, *not the current location*, *distinct*) on every tool argument of 6,213 archived AutoTool steps and keep templates decidable on at least 20 accepted steps and Satisfied on at least 90% of them (25 constraints over 12 tools). **LLM-written.** In one zero-shot call at temperature 0, the agent's LLM selects the preconditions of each tool from the same library, given only the official tool documentation (50 constraints over 25 tools); it chooses and binds predicates, and the grounding is ours.

3.3 EXACT BOUNDED REPAIR

CRA computes Eq. 2 by depth-first search over the candidate product in rank order, pruning a partial tuple as soon as a check is Violated; because every check is monotone, pruning never discards a certified tuple (Proposition 3 and Algorithm 1, Appendix A). The search is exact within a budget of 256 tuples or 50 ms, which it never reached, and a binding call takes 5.7 ms at the median on one CPU core. In all 204 ScienceWorld repairs of the expert catalog, the first certified tuple is also the highest-scoring one, so searching past the first certified tuple adds nothing.

4 EVALUATION PROTOCOL

Agents and environments. We use AutoTool (Jia & Li, 2026) (commit `c7010ff`) in AgentBoard's ScienceWorld and ALFWorld environments (Wang et al., 2022; Shridhar et al., 2021; Ma et al., 2024), driven by Gemini 3.8 Flash (Gemini Team, Google, 2023) through an OpenAI-compatible endpoint (alias `gemini-3.8-flash-high`, temperature 0, accessed September 2026; no fixed snapshot) and the official SimCSE-RoBERTa-base tool predictor (Gao et al., 2021); ALFWorld's closed loop and branches are repeated with Claude Sonnet 4.6 (alias `claude-sonnet-4-6`). AutoTool acts model-free when its tool predictor's confidence exceeds a threshold τ and its filler completes every argument, and it allows at most c consecutive model-free steps. Episodes use the simulator's task description and end on completion, a negative ScienceWorld score, or 30 steps. All gates wrap AutoTool's parameter filler; AutoTool's source is not modified.

Counterfactual execution. We take development logs, 445 ScienceWorld and 938 ALFWorld episodes of the gated and native agents under AutoTool's default τ=0.2, c=1, and window W=2, rebuild the candidate lists and the evidence at every step from the logged prefix, and keep every decision point: a step whose logged tool has at least two arguments that the cache can fill. This gives 563 ScienceWorld points (54 tasks; 289 `move`, 223 `use`, 51 `pour`) and 1776 ALFWorld points (134 tasks; 980 `take`, 500 `put`, 296 `clean`, `heat`, or `cool`). Because every gate receives the logged tool, the study stress-tests argument binding rather than sampling the states at which the agent would skip the model. For each gate we compute the tuple it would execute, restore the simulator by replaying the logged prefix (accepted only if every observation, and in ScienceWorld every score, matches the log), execute the tuple (rendering `heat` and `cool` with their own verbs; Appendix C), and label it *refused* if the observation is a failure message of the simulator: one of 38 messages extracted from ScienceWorld's action handlers (e.g., *you can't move something into itself*) or ALFWorld's *nothing happens* (Appendix C). A diverging replay is retried once and otherwise undecidable; tuples identical to the logged action reuse its logged outcome. We report gates on the *common states* at which every gate's outcome is decidable, 410 ScienceWorld points (41 tasks) and all ALFWorld points; counting the undecidable ScienceWorld outcomes as accepted or as refused bounds the refusal rate of CRA to [25.0, 35.3]%, below the lower bound of every gate without certificates (Appendix D). CIs are 95% percentile intervals from 10,000 task-bootstrap resamples.

Gates compared. Besides G_{U}, G_{R}, and CRA with each certificate source, *distinct repair* takes per slot the highest-ranked candidate whose canonical name differs from earlier slots without any other check, and an *Unknown-as-pass* CRA reads Unknown as Satisfied, i.e., fails open. To test whether a model call is the better check, *LLM select* gives the agent's LLM, in one short call, the goal, the last three steps, the tool documentation, and the same K=8 candidates per slot, and asks it to pick a tuple or answer ASK (prompt in Appendix B). As references we report the refusal rate of the LLM's own logged actions at the same points and, where the logged action came from the model, whether a gate's action equals it.

Closed loop. To measure whole-episode effects, we run the agent under an *aggressive* setting (τ=0.1, c=3; AutoTool's cap of 30% model-free steps per episode is kept). On the 58 ScienceWorld *development tasks* whose types require a multi-argument tool, five arms run: unmodified AutoTool with its native filler, and our shared filler unchecked (G_{U}), gated by G_{R} (expert catalog), or gated by CRA with the expert or the LLM-written catalog. The *held-out tasks* take as many per type from unused test variations (Appendix C); that run keeps native AutoTool, the unguarded filler, and CRA (expert), and adds to the latter two a rule that sends irreversible actions to the model, registered with its outcomes before the run; later registered runs gate AutoTool's own filler by G_{R} and CRA (35 development tasks where it executed uncertified model-free actions) and add a fail-closed and, in ALFWorld, an effect-aware CRA (Section 5.4). On all 134 ALFWorld test games, native AutoTool, the unguarded filler, the reject gate, and CRA run once with each model. Arms run concurrently against the same endpoint; we report per-episode means with paired task-bootstrap intervals and, for success, exact McNemar tests; where few episodes differ, we read the test rather than the interval.

5 Results

5.1 How do model-free actions fail?

Mismatched values. With a two-record window the most recent value ranks first for every slot, so the greedy proposal put the same entity into several slots at all 563 ScienceWorld decision points (`move(thermometer, thermometer)`) and at 92.7% of the 1776 ALFWorld points (`take(countertop 1, countertop 1)`). At the common points (Section 4) these proposals are refused 80.2% and 93.7% of the time, the model's own actions 24.7% and 0.2%.

Stale values. In our ScienceWorld development logs the environment refused 51% of 1,206 model-free actions but 29% of the model's own, mostly navigation to the room just entered (552 of 553; the model: 31 of 1,541), as in ALFWorld. We then added *not the current location* to the expert catalog and the template library (Appendix C).

Table 1: **Gates at the multi-argument decision points** of ScienceWorld (410 common points, 41 tasks) and ALFWorld (1776 points, 134 tasks) of the development logs, by counterfactual execution: share of points at which a gate acts without a full model call, environment-accepted (Acc.) and refused (Ref.) executed actions, and refusal rate Ref./(Acc.+Ref.) with task-bootstrap 95% CI (*: none refused, exact one-sided 95% bound; –: undefined). *LLM select* spends one short model call per point. Model's own actions: at the common points whose logged action came from the model.

	ScienceWorld				ALFWorld			
Gate	Acts %	Acc.	Ref.	Refused %	Acts %	Acc.	Ref.	Refused %
Unguarded	100.0	81	329	80.2 [68.6, 92.5]	100.0	112	1664	93.7 [91.9, 95.4]
Reject (expert catalog)	0.0	0	0	–	2.5	37	8	17.8 [6.5, 33.3]
Distinct repair	90.7	129	243	65.3 [56.3, 77.1]	39.0	112	581	83.8 [79.1, 88.0]
Unknown-as-pass CRA	72.9	137	162	54.2 [45.7, 65.8]	99.8	112	1661	93.7 [91.8, 95.4]
CRA, expert catalog	43.7	128	51	28.5 [21.8, 41.4]	13.3	174	63	26.6 [15.9, 36.7]
CRA, mined catalog	65.4	108	160	59.7 [51.9, 68.5]	100.0	112	1664	93.7 [91.9, 95.4]
CRA, LLM-written catalog	47.8	124	72	36.7 [30.5, 46.9]	0.0	0	0	–
LLM select (short call)	27.8	109	5	4.4 [1.1, 8.9]	17.3	308	0	0.0 [0, 1.0]*
Model's own actions	–	283	93	24.7 [18.5, 32.9]	–	1754	4	0.2 [0.1, 0.5]

5.2 RQ1: Do certificates make model-free actions executable?

Certification cuts refusals by about two thirds. In ScienceWorld (Table 1) the unguarded gate is refused 80.2% of the time and the reject gate never acts, because every greedy proposal fails the expert catalog. CRA acts at 43.7% of the points and is refused 28.5% of the time, 51.8 points below the unguarded gate (CI [43.7, 60.4]). ALFWorld reproduces the pattern (26.6% against 93.7%, CI of the difference [57.2, 77.1]), while the reject gate acts at 2.5% of the points. Each accepted action that CRA adds to rejection's (128 and 137) saves a call (Proposition 1). The reduction is not only abstention: where CRA acts, the unguarded gate is refused 56.4% of the time (27.9 points more, CI [19.5, 42.6]). On held-out tasks, unused in building the tracker, the frozen catalog agrees: CRA is refused 21.1% of the time against 82.6% (CI of the difference [44.3, 88.2]; 86 states, 20 tasks).

Repair needs certificates. Repairs without certificates accept about as many actions but are refused far more often: distinct repair and Unknown-as-pass are refused 36.8 and 25.7 points more often than CRA in ScienceWorld (CIs [29.7, 44.9] and [19.3, 32.5]), and distinct repair 83.8% of the time in ALFWorld. Among the ingredients (Table 14, on its 437 common points), not passing on missing evidence matters most: Unknown-as-pass accepts about as many actions (140 vs. 131) but is refused 164 times instead of 51. Of CRA's 51 ScienceWorld refusals, 46 violate preconditions the catalog does not encode (*I'm not sure how to use the stove*) and 5 are grounding errors. In ALFWorld, adding one missing predicate (*the agent is at the receptacle*) turns all 55 refused `take` actions of CRA into asks (Appendix D). With the LLM-written catalog, CRA is refused 36.7% of the time in ScienceWorld, 8.2 points above the expert catalog (CI [1.2, 12.8]); the mined catalog is close to distinct repair, and neither generic catalog transfers to ALFWorld (Appendix B).

A model call as the check. LLM select is the most reliable gate (refused 4.4% and 0.0% of the time) but acts less often (at 27.8% and 17.3% of the points; CRA: 43.7% and 13.3%); counting its fallback calls, it spends 78% and 101% of a full call per point, CRA 56% and 87% (Appendix D).

Answer to RQ1. Largely, with an expert catalog: without a model call, CRA cuts refusals by about two thirds and acts where rejection cannot. Uncertified repair does not, nor do generic catalogs in ALFWorld.

5.3 RQ2: Is an executable action the right one?

Repair replays the recent past (Figure 2). At the common points where the model itself acted, 51 of the 99 ScienceWorld actions of CRA that the environment accepts differ from the model's action (51.5%), against 21.5% for LLM select. Repair can only reach cached values, and the model's own tuple is certifiable from the cache at only 15.4% of these 376 points. What repair reaches instead is the agent's recent past: under recency ranking the best certified tuple is often the previous call, whose catalogued preconditions still hold after its effect has happened. Separating re-read

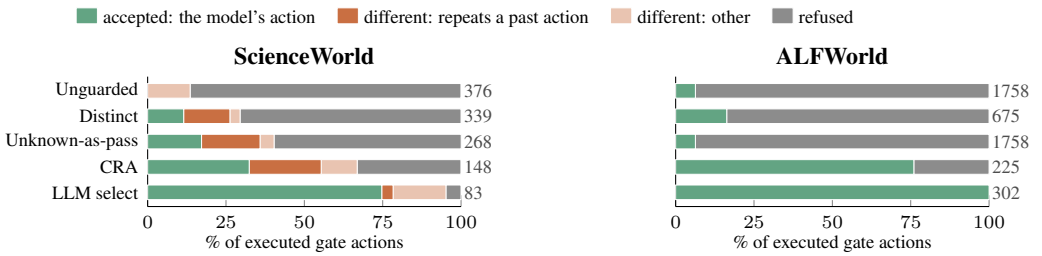


Figure 2: **Executable is not the same as right.** Executed gate actions (CRA: expert catalog) at the common points where the model acted (376 ScienceWorld, 1758 ALFWorld; right: executed actions): accepted and equal to the model's action, accepted but different (dark: repeating an earlier action), or refused. Half of what CRA gets accepted in ScienceWorld is not the model's action, mostly a repeat.

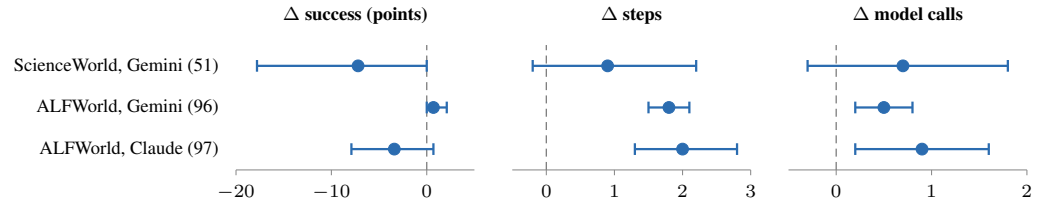


Figure 3: **Same-state branches: going back costs more than it saves.** Δ: CRA's branch minus the model's, both continued by the same LLM without model-free actions (three runs each; states in parentheses), with 95% task-bootstrap CIs; rates and tests in Table 6.

measurements from state changes (Table 13), 35 of CRA's 99 accepted actions re-execute a state change already made in the episode, such as *move ice to metal pot* once more, against 11 of the model's 283 (difference in share 31.5 points, task-clustered CI [19.2, 51.1]), while both often re-read a thermometer (40 and 72); where AutoTool itself attempted a model-free fill, 33 of 35 accepted actions of CRA repeat the previous one, none of 34 unguarded ones. ALFWorld shows the same mechanism in navigation, which the counterfactual study, restricted to multi-argument decisions, excludes: every repaired navigation that the environment accepted in closed loop went back to the previous location (337 with Gemini, 436 with Claude), against 8% and 5% of the model's own accepted navigations in the reject arm. Nor is replay an artifact of our filler or ranking: on AutoTool's own filler, 14 of 18 accepted repairs of CRA repeat an earlier action, and excluding repeats raises its refusal rate from 28.5% to 84.3% (Appendix D).

Same-state branches. Agreement with one logged action is not task success, so we branch from single states (Figure 3; Table 6; protocols in Table 7): the 51 ScienceWorld points at which the accepted action of CRA differs from the model's, and the first navigation per ALFWorld game that CRA repaired and the environment accepted in closed loop (96 games with Gemini, 97 with Claude), where rejection would have asked the model. One branch executes CRA's action, the other the model's; the same LLM then continues both without model-free actions, three times each.

Going back costs more than it saves. In ALFWorld the model's own next action is a navigation in only 24% and 51% of the runs and never to the location CRA chose; otherwise it acts where the agent stands (*take*, *put*, *open*). Going back costs 1.8 and 2.0 extra steps with Gemini and Claude (CIs [1.5, 2.1] and [1.3, 2.8]) and 0.5 and 0.9 extra model calls (CIs [0.2, 0.8] and [0.2, 1.6]), although the repaired step itself needed none; success does not differ detectably. In ScienceWorld the model's own action succeeds numerically more often, against noisy outcomes (a branch rerun from the same state changes its outcome at 35% of the states): the branches succeed in 56.2% and 49.0% of the runs, 7.2 points apart (task-clustered CI [0.0, 17.8]), and 10 of 17 tasks favor the model's branch against 5 (sign test p=0.302). The registered first-run analysis points the same way (13.7 points, CI [0.0, 29.5]; 10 vs. 3 discordant states, exact McNemar p=0.092 treating states as independent).

Table 2: **Closed loop under aggressive reuse** (τ=0.1, c=3). Per-episode means, except totals for *Multi/Nav.* (model-free multi-argument/`go_to` actions, refused) and *Fail* (negative-score endings, by a model-free `focus on`); *MF*: model-free actions (refused share; bottom: total, refused); *ask irrev.*: irreversible actions go to the model; *fail-closed*: every tool without a domain predicate goes to the model; *effect-aware*: a model-free action that repeats an accepted action other than an observation goes to the model (†later runs). Bottom: ALFWorld, per model.

Arm	Succ. (%)	Score	Calls	Tok. (k)	MF (ref.)	Multi (ref.)	Nav. (ref.)	Fail (focus)
ScienceWorld, development tasks (58)								
AutoTool, native filler	46.6	61.1	17.8	90.2	4.1 (41%)	20 (20)	71 (45)	6 (3)
Shared filler, unguarded	44.8	73.5	19.0	112.3	4.1 (50%)	29 (26)	66 (66)	3 (1)
+ Reject (expert)	39.7	45.2	19.3	105.7	2.6 (9%)	0 (0)	0 (0)	12 (6)
+ CRA (expert)	43.1	58.5	18.6	100.3	2.9 (7%)	40 (4)	0 (0)	7 (5)
+ CRA (LLM-written)	41.4	74.3	21.2	117.1	2.5 (14%)	18 (12)	0 (0)	3 (0)
ScienceWorld, held-out tasks (57)								
AutoTool, native filler	35.1	53.8	17.8	93.9	4.0 (42%)	9 (9)	92 (57)	8 (4)
Shared filler, unguarded	43.9	59.3	18.5	89.2	4.4 (54%)	34 (16)	94 (94)	7 (2)
+ ask irrev.	38.6	59.8	19.3	102.0	3.7 (56%)	32 (12)	82 (82)	5 (0)
+ CRA (expert)	31.6	42.5	19.3	107.2	2.9 (14%)	19 (4)	0 (0)	11 (6)
+ CRA, ask irrev.	36.8	55.8	19.7	106.5	1.9 (10%)	21 (4)	0 (0)	7 (0)
+ CRA, fail-closed†	24.6	32.9	20.3	151.4	0.8 (23%)	21 (9)	0 (0)	10 (0)

ALFWorld, test games (134)	Gemini 3.8 Flash				Claude Sonnet 4.6			
Arm	Succ. (%)	Calls	Steps	MF (ref.)	Succ. (%)	Calls	Steps	MF (ref.)
AutoTool, native filler	99.3	8.3	8.4	11 (4)	89.6	12.3	12.8	55 (3)
Shared filler, unguarded	97.8	8.6	12.2	475 (460)	84.3	11.6	16.7	628 (548)
+ Reject (expert)	100.0	8.3	8.4	28 (17)	88.1	11.9	13.9	245 (119)
+ CRA (expert)	97.0	8.9	12.3	456 (111)	76.9	12.7	17.8	658 (147)
+ CRA, effect-aware†	99.3	8.2	9.1	122 (104)	86.6	12.5	14.9	305 (169)

In ScienceWorld the tool is the model's and repair changes its arguments; in ALFWorld the model often chooses a different tool, and repair goes back instead, spending the call it saved and more.

Answer to RQ2. Often not. Preconditions say when an action *can* run, not whether it still *needs* to, so recency-ranked repair replays the agent's recent past with both fillers we tested; from the same states, such actions cost ALFWorld agents steps and calls.

5.4 RQ3: What does gating do to whole episodes?

ScienceWorld. On the development tasks (Table 2, top), CRA changes success by -1.7 points relative to the unguarded filler (CI [-15.5, 13.8]). CRA removes most refused model-free actions (12 of 171; unguarded: 121 of 240). Fewer refusals do not mean fewer calls: CRA changes them by -0.3 per episode (CI [-2.0, 1.3]). No ScienceWorld success difference is significant (Table 9), and single runs measure the gates imprecisely: in 54 of 58 tasks the paired episodes decouple before the gates first decide differently, as the model acts differently after identical observations (Appendix D). On AutoTool's own filler (Table 11), CRA succeeds in 34.3% of the episodes and rejection in 17.1% (7 vs. 1 tasks, p=0.070), but all 35 pairs decoupled before the gates differed.

Irreversible actions end most failures. Most failures (negative-score endings) of expert-gated arms end with a model-free `focus on` that names the current room, certified on record checks for lack of a predicate (5 of CRA's 7). On held-out tasks, sending `focus on`, `eat`, and `mix`, or every uncatalogued tool (*fail-closed*), to the model removes these endings (-10.5 and -10.5 points) at 0.4 and 1.0 extra calls per episode, but not the failures, as the model then focuses wrongly itself (7 and 10 episodes; 4 without the rules); fail-closed changes success by -7.0 points (CI [-21.1, 7.0]).

ALFWorld with two models. Nearly all model-free actions of the unguarded filler are refused (97% with Gemini, 87% with Claude; Table 2, bottom), mostly navigation to the current location, which CRA repairs by going back. Relative to the reject gate, which almost never navigates without the

model, CRA takes 3.8 and 3.9 more steps per episode and changes success by -3.0 and -11.2 points (CIs [-6.0, -0.7] and [-17.2, -6.0]; Table 10); with Gemini only 4 games differ, all lost by CRA (exact McNemar p=0.125), with Claude CRA loses 16 and wins 1 (p <0.001). Unlike in ScienceWorld, paired games mostly stay coupled until the gates decide differently (3 of 4 and 12 of 17 discordant games). Asking instead of repeating removes the loss: an effect-aware CRA that asks the model rather than repeat an accepted action other than an observation (a later registered run) changes success relative to CRA by 9.7 points with Claude (15 vs. 2 games, p=0.002) and 2.2 with Gemini (4 vs. 1), and by -1.5 and -0.7 relative to rejection.

Answer to RQ3. Removing refusals brings no detectable gain in success or calls over unguarded reuse or native AutoTool. ScienceWorld episodes decouple through the nondeterministic model before gates differ; in ALFWorld, repair lowers success relative to rejection, most clearly with Claude, and asking instead of repeating restores it. What hurts, a step back or an irreversible `focus on`, passes executability checks (Table 15).

6 Related work

Cost-efficient LLM agents. Tool-using agents (Yao et al., 2023; Qin et al., 2024; Patil et al., 2023) spend most of their budget on model calls, which semantic caches (Bang, 2023), model cascades (Chen et al., 2023), decoupled or parallel planners (Xu et al., 2023; Kim et al., 2024), and runtime supervisors (Lin et al., 2026) reduce; AutoTool (Jia & Li, 2026) skips calls by predicting the next tool and its arguments from usage graphs. Speculative actions (Ye et al., 2026; Guan et al., 2026) check a fast guess against the slow model, saving latency, and vCache (Gaspar Schroeder et al., 2026) bounds the error of cached responses. These methods decide *when* to skip or consult the model; we study *what* may be executed once it is skipped.

Checking actions. Guardrails and policy verifiers (Inan et al., 2023; Chen et al., 2025; Winston et al., 2026), emulators and benchmarks (Ruan et al., 2024; Yao et al., 2024; Debenedetti et al., 2024), self-correction (Shinn et al., 2023; Gou et al., 2024; Chen et al., 2024), and constrained decoding (Geng et al., 2023; Willard & Louf, 2023) check, expose, correct, or restrict actions that a model produces, usually with further calls; counterfactual replay also attributes failures in agent traces (Zhang et al., 2026). CRA instead repairs actions that no model generated: it is a shield (Alshiekh et al., 2018) with observation-grounded preconditions that abstains on inconclusive evidence, like runtime verification (Bauer et al., 2011) and selective prediction (Chow, 1970; Geifman & El-Yaniv, 2017; Kamath et al., 2020), and repairs by finite-domain constraint satisfaction (Mackworth, 1977; Dechter, 2003). Precondition checks and plan repair are classical in planning (Fikes & Nilsson, 1971; McDermott et al., 1998; Fox et al., 2006) and LLM planning (Liu et al., 2023; Ahn et al., 2022), whose operators also carry effects, which our certificates ignore and which separate progress from repetition. Asking before irreversible actions echoes reversibility-aware RL (Eysenbach et al., 2018; Grinsztajn et al., 2021) and LLM planners that ask for help (Ren et al., 2023).

7 Discussion and conclusion

Can an agent skip the model without skipping the checks? Partly. Where the catalog has predicates, grounded three-valued certificates cut refusals of model-free actions by about two thirds without a model call. They cannot tell whether an action is still needed: catalogued preconditions still hold after an action has taken effect, so recency-ranked repair replays the agent's recent past, which from the same states cost ALFWorld agents steps and calls. ScienceWorld's fatal actions came from uncatalogued tools; asking the model there removed them, not the failures. In practice: certify before executing, add predicates where refusals concentrate, and ask rather than repeat an accepted state change, which in ALFWorld recovered the success that repair lost, matching rejection.

Limitations. Predicates and grounding are hand-built per environment; our gates mostly wrap a shared recency-ranked filler (AutoTool's own only on 35 closed-loop tasks). ScienceWorld's counterfactual states also served to build the tracker, and ALFWorld's closed loop reuses its development games (Appendix C); closed-loop comparisons rest on 57–134 tasks per arm, one agent framework, and nondeterministic endpoints, partly across days; counterfactual execution needs a simulator restorable by replay.

Reproducibility statement

Section 4 and Appendix C specify the agent, both simulators, the model endpoints, task lists (the held-out list is regenerated by a script from ScienceWorld's test variations), decision-point definition, restoration and divergence rules, refusal labels, the restoration control, and statistics. Appendix B lists all three catalogs, the instruction and predicate library of the prompt used to obtain the LLM-written one, how the tool documentation is appended to it, and the LLM-select prompt. The code of the gate, the counterfactual-execution pipeline, the analysis scripts, the per-step logs, and the prompts will be released upon publication. The LLM endpoints have no fixed snapshot and are not deterministic at temperature 0, so closed-loop, branch, and LLM-select numbers may drift; we release every LLM-select response and every branch run with its states, actions, and outcomes. Counterfactual execution of the gates involves no model call apart from LLM select; we release the logged trajectories together with every execution record and refusal label, from which all counterfactual numbers can be recomputed, while restoring states anew is subject to the same replay divergence as in our runs.

Ethics statement

This work uses simulated text environments and involves no human subjects or personal data. Gating model-free actions is a safety measure: it prevents a cost-saving component from executing reused actions that fail the catalogued checks. Certification is only as good as its predicates; in deployments where actions have side effects, uncatalogued preconditions remain a risk that a gate of this kind reduces but does not remove, and irreversible actions should be decided by the model or a human rather than by reuse.

AI use statement

The agent under study is driven by an LLM (Gemini 3.8 Flash, and Claude Sonnet 4.6 in the ALFWorld closed loop and its branches); one of the three certificate catalogs is written by that LLM, and the LLM-select baseline queries it. These uses are part of the method and are described in the paper. In addition, the authors used LLM-based assistants to implement and modify code and to draft and edit the text; all reported numbers are produced by scripts from logged runs. The authors take full responsibility for the content of this paper.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, et al. Do as I can, not as I say: Grounding language in robotic affordances. In *Proc. CoRL*, 2022.

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

Glenn Ammons, Rastislav Bodík, and James R. Larus. Mining specifications. In *Proc. POPL*, pp. 4–16, 2002.

Fu Bang. GPTCache: An open-source semantic cache for LLM applications enabling faster answers and cost savings. In *Proc. Workshop for Natural Language Processing Open Source Software (NLP-OSS)*, 2023.

Andreas Bauer, Martin Leucker, and Christian Schallhart. Runtime verification for LTL and TLTL. *ACM Transactions on Software Engineering and Methodology*, 20(4):14:1–14:64, 2011.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *arXiv:2305.05176*, 2023.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. In *Proc. ICLR*, 2024.

Zhaorun Chen, Mintong Kang, and Bo Li. ShieldAgent: Shielding agents via verifiable safety policy reasoning. In *Proc. ICML*, volume 267, pp. 8313–8344, 2025.

C. K. Chow. On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, 16(1):41–46, 1970.

Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *Proc. NeurIPS Datasets and Benchmarks Track*, 2024.

Rina Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

Michael D. Ernst, Jeff H. Perkins, Philip J. Guo, Stephen McCamant, Carlos Pacheco, Matthew S. Tschantz, and Chen Xiao. The Daikon system for dynamic detection of likely invariants. *Science of Computer Programming*, 69(1–3):35–45, 2007.

Benjamin Eysenbach, Shixiang Gu, Julian Ibarz, and Sergey Levine. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. In *Proc. ICLR*, 2018.

Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3–4):189–208, 1971.

Maria Fox, Alfonso Gerevini, Derek Long, and Ivan Serina. Plan stability: Replanning versus plan repair. In *Proc. ICAPS*, pp. 212–221, 2006.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. SimCSE: Simple contrastive learning of sentence embeddings. In *Proc. EMNLP*, pp. 6894–6910, 2021.

Luis Gaspar Schroeder, Aditya Desai, Alejandro Cuadron, Kyle Chu, Shu Liu, Mark Zhao, Stephan Krusche, Alfons Kemper, Matei Zaharia, and Joseph E. Gonzalez. vCache: Verified semantic prompt caching. In *Proc. ICLR*, 2026.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Proc. NeurIPS*, 2017.

Gemini Team, Google. Gemini: A family of highly capable multimodal models. *arXiv:2312.11805*, 2023.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. Grammar-constrained decoding for structured NLP tasks without finetuning. In *Proc. EMNLP*, pp. 10932–10952, 2023.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *Proc. ICLR*, 2024.

Nathan Grinsztajn, Johan Ferret, Olivier Pietquin, Philippe Preux, and Matthieu Geist. There is no turning back: A self-supervised approach for reversibility-aware reinforcement learning. In *Proc. NeurIPS*, 2021.

Yilin Guan, Qingfeng Lan, Fei Sun, Dujian Ding, Devang Acharya, Chi Wang, William Yang Wang, and Wenyue Hua. Dynamic speculative agent planning. In *Proc. ICLR*, 2026.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. Llama Guard: LLM-based input-output safeguard for human-AI conversations. *arXiv:2312.06674*, 2023.

Jingyi Jia and Qinbin Li. AutoTool: Efficient tool selection for large language model agents. In *Proc. AAAI*, 2026. doi: 10.1609/aaai.v40i37.40389.

Amita Kamath, Robin Jia, and Percy Liang. Selective question answering under domain shift. In *Proc. ACL*, 2020.

Sehoon Kim, Suhong Moon, Ryan Tabrizi, Nicholas Lee, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. An LLM compiler for parallel function calling. In *Proc. ICML*, 2024.

Stephen Cole Kleene. *Introduction to Metamathematics*. North-Holland, Amsterdam, 1952.

Fulin Lin, Shaowen Chen, Ruishan Fang, Hongwei Wang, and Tao Lin. Stop wasting your tokens: Towards efficient runtime multi-agent systems. In *Proc. ICLR*, 2026.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. LLM+P: Empowering large language models with optimal planning proficiency. *arXiv:2304.11477*, 2023.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. AgentBoard: An analytical evaluation board of multi-turn LLM agents. In *Proc. NeurIPS*, 2024.

Alan K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118, 1977.

Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. PDDL – the planning domain definition language. Technical Report CVC TR-98-003, Yale Center for Computational Vision and Control, 1998.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. *arXiv:2305.15334*, 2023.

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. In *Proc. ICLR*, 2024.

Allen Z. Ren, Anushri Dixit, Alexandra Bodrova, Sumeet Singh, Stephen Tu, Noah Brown, Peng Xu, Leila Takayama, Fei Xia, Jake Varley, Zhenjia Xu, Dorsa Sadigh, Andy Zeng, and Anirudha Majumdar. Robots that ask for help: Uncertainty alignment for large language model planners. In *Proc. CoRL*, 2023.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. Identifying the risks of LM agents with an LM-emulated sandbox. In *Proc. ICLR*, 2024.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Proc. NeurIPS*, 2023.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Proc. NeurIPS*, 2023.

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

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. ScienceWorld: Is your agent smarter than a 5th grader? In *Proc. EMNLP*, 2022.

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

Cailin Winston, Claris Winston, and René Just. Solver-aided verification of policy compliance in tool-augmented LLM agents. *arXiv:2603.20449*, 2026.

Binfeng Xu, Zhiyuan Peng, Bowen Lei, Subhabrata Mukherjee, Yuchen Liu, and Dongkuan Xu. ReWOO: Decoupling reasoning from observations for efficient augmented language models. *arXiv:2305.18323*, 2023.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *Proc. ICLR*, 2023.

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

Naimeng Ye, Arnav Ahuja, Georgios Liargkovas, Yunan Lu, Kostis Kaffes, and Tianyi Peng. Speculative actions: A lossless framework for faster AI agents. In *Proc. ICLR*, 2026.

Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng Yan. AgenTracer: Who is inducing failure in the LLM agentic systems? In *Proc. ICLR*, 2026.

A ALGORITHM AND PROOFS

Algorithm 1 Certify–repair–ask for one decision point

Require: tool T; ranked candidate lists $V_1, \dots, V_m$; check set Φ_T; evidence E_t; budget of N tuples and Δ ms (N=256, Δ=50)

if $\Phi_T(g_t) = \mathsf{S}$ **then return** g_t ▷ certify: identical to rejection
end if
$a^* \leftarrow \bot$ with $s(\bot) = -\infty$; explored $\leftarrow 0$
procedure DFS($a_{1:k}$) ▷ repair
  if some $\phi \in \Phi_T$ is V on $a_{1:k}$ **then return**
  end if
  if $k = m$ **then**
    explored $\leftarrow$ explored $+1$; **if** explored $> N$ or elapsed $> \Delta$ **then return** $\bot$ for the whole search
    if $\Phi_T(a_{1:m}) = \mathsf{S}$ and $s(a_{1:m}) > s(a^*)$ **then** $a^* \leftarrow a_{1:m}$
    end if
    return
  end if
  for $v \in V_{k+1}$ in rank order **do** DFS($a_{1:k} \,\|\, v$)
  end for
end procedure
DFS($\emptyset$)
return a^* ▷ ask the LLM if no certified tuple exists

Throughout, the repair search enumerates $V_1 \times \cdots \times V_m$ depth-first in rank order and replaces its incumbent only on a strictly higher score, so ties are resolved in favor of the tuple visited first; the greedy proposal g_t is visited first and maximizes s over the product.

Proof of Proposition 1. If $G_{\mathrm{R}}(t) \neq \bot$ then $\Phi_T(g_t) = \mathsf{S}$, so $g_t \in \mathcal{C}_t$. Since g_t maximizes s over the whole product and is visited first, it is the maximizer over $\mathcal{C}_t \subseteq V_1 \times \cdots \times V_m$ that the search returns (Algorithm 1 even returns it before searching). Hence $G_{\mathrm{CRA}}(t) = g_t = G_{\mathrm{R}}(t)$. By Proposition 3, a search that finishes within budget visits every certified tuple, so it returns $\bot$ exactly when $\mathcal{C}_t = \emptyset$. The set of decision points at which G_{R} acts is therefore contained in the set at which G_{CRA} acts, and the difference consists of the points with $g_t \notin \mathcal{C}_t$ and $\mathcal{C}_t \neq \emptyset$. □

Proposition 2 (Soundness is inherited)**.** *Call ϕ_j sound for a precondition p_j if $\phi_j(a) = \mathsf{S}$ implies that $p_j(a)$ holds in the true environment state. If every check in Φ_T is sound, every tuple executed by G_{R} or G_{CRA} satisfies all preconditions in the catalog, so, if every refusal is explained by a violated precondition, it stems from one that the catalog does not encode. A gate that reads U as S loses this guarantee as soon as evidence is missing.*

Proof of Proposition 2. Both gates execute only tuples with $\Phi_T(a) = \mathsf{S}$. Under strong Kleene conjunction, $\Phi_T(a) = \mathsf{S}$ holds if and only if $\phi_j(a) = \mathsf{S}$ for every j, and soundness gives $p_j(a)$ for every j. If the environment refuses a, the refusal therefore violates some precondition that is not among $\{p_j\}$. A gate that reads Unknown as Satisfied also executes tuples with $\Phi_T(a) = \mathsf{U}$, in which some $\phi_j(a) = \mathsf{U}$; soundness says nothing about $p_j(a)$ in that case, so the refusal may violate a catalogued precondition. □

Proposition 3 (Exact pruning)**.** *If every check is monotone, i.e., once V on a partial tuple it is V on all its extensions, then depth-first search over $V_1 \times \cdots \times V_m$ that discards a partial tuple as soon as a check is V returns $G_{\mathrm{CRA}}(t)$ exactly.*

Table 3: Three sources of certificates, evaluated with the same grounded state tracker. *Archived replay*: 6,213 logged AutoTool steps; model calls of the reject gate (R) and of CRA (C), and how often the catalog flags the logged action as Violated when the environment accepted it (false alarm) or refused it (detection); the gate also asks on Unknown, so false alarms understate how often it blocks accepted actions. The mined catalog is cross-fitted over two episode folds. *Counterfactual execution*: CRA with each catalog on the 410 common ScienceWorld decision points (Section 4). Expert counts exclude the ALFWorld `go` check added later (Appendix C).

				Archived replay			Counterfactual	
Source	Input	Tools	Rules	R→C calls	False al. ↓	Detect. ↑	Skip	Refused
Expert	domain knowledge	16	19	2312→1341	4.3%	23.0%	43.7%	28.5%
Mined	6,213 logged steps	12	25	2079→1447	1.0%	5.9%	65.4%	59.7%
LLM-written	tool documentation	25	50	3192→2702	6.8%	31.9%	47.8%	36.7%

Proof of Proposition 3. The search discards exactly the completions of partial tuples on which some check is V. By monotonicity every such completion is V on that check, hence $\Phi_T \neq \mathsf{S}$, so no certified tuple is discarded. Unknown never prunes. Every certified tuple is therefore visited and scored, and the incumbent after the search is the maximizer over $\mathcal{C}_t$, or $\perp$ if $\mathcal{C}_t = \emptyset$. In our implementation each check returns Unknown until all of the slots it reads are filled and depends only on those slots, which makes it monotone. □

B CERTIFICATE CATALOGS

Table 3 compares the three catalogs on the archived replay and in counterfactual execution. On the archived replay, every catalog lets CRA save calls that rejection pays, but the mined catalog detects the fewest refusals (5.9%), because mining favors predicates that hold almost always, and the LLM-written one detects the most (31.9%) at a false-alarm rate close to the expert's (6.8% vs. 4.3%). The catalogs differ in expressiveness as well as in source (the expert's checks are composite and include some that the template library lacks), and all three share our grounding, so the comparison is between complete configurations rather than between sources in isolation.

Grounding rules. A feedback sentence, a description variant, or an animal can pass a string test, so predicates must be grounded in what the agent has observed. We maintain four kinds of evidence from the observation history, compare names only after canonicalization (never by substring), and update evidence only from accepted actions. *Identity*: equal canonical names (articles and *substance called* dropped) denote one entity; names that differ only by a description clause (*fountain* vs. *fountain (containing water)*) or a location qualifier are unresolved (Unknown). *Presence*: a value is present if its canonical name is visible or carried; a qualified referent "x in y" is present only if x is carried or observed inside an accessible y, and Unknown if y is accessible but its recorded contents omit x. *Role*: a destination is a container only if an observation states containment, support, or a door; visibility is not evidence. *Location*: the current location is read from room descriptions and successful moves, and a navigation target that names it is Violated. The latest room listing defines the visible set, pick-up and put-down move objects between the visible and carried sets, and a container's latest listing defines its contents; a plain name absent from both sets is Violated, which is sound given unique canonical names and a complete room listing, and everything undecided is Unknown. Because liquid state is not observable in text, the expert catalog keeps pouring and dunking predicates Unknown, so CRA with it always asks before those actions.

Expert catalog. Table 4 lists the per-tool domain predicates of the expert catalog; every tool additionally carries record-validity and schema-type checks. Two synthetic record tools used only in unit tests are omitted.

Mined and LLM-written catalogs. Table 5 lists both catalogs. The mined catalog keeps a template on a tool argument if it is decidable on at least 20 accepted archived steps and Satisfied on at least 90% of them. The LLM-written catalog is the output of one call with the prompt below, after the 6 constraints that refer to argument names of the identically named ALFWorld tools `open`,

Table 4: Expert catalog (domain predicates per tool). Rules apply only in their own environment; the ScienceWorld `use` check, for instance, does not apply to ALFWorld's one-argument `use`.

Environment	Tool	Domain predicates
ScienceWorld	`move`	source present; destination is a container; distinct
	`use`	tool and target present; distinct
	`pour`, `dunk`	present; distinct; liquid state (always Unknown)
	`pick_up`	object present
	`go_to`	atomic string; known location; not the current location
ALFWorld	`take`	object at the receptacle
	`put`	object carried
	`heat`, `cool`, `clean`	object and receptacle co-accessible
	`go`	not the current location (added before the ALFWorld closed loop)
ToolQuery-Acad.	`neighbourCheck`	node in graph; graph loaded
	`author/paperNodeCheck`	node type
	`author/paperEdgeCheck`	both nodes of the expected type

`close`, `use`, and `examine` are dropped in favor of the ScienceWorld tools of the same name; this rule was fixed before any evaluation.

The prompt below describes *visible* as the listing of the current room; in our tracker the visible set also contains carried objects, so *visible* and *reachable* coincide and *in inventory* is the only predicate that singles out carried objects.

Prompt for the LLM-written catalog. The prompt consists of the instruction and predicate library below, followed by one line per tool with its name, arguments, description, and argument descriptions, copied from AutoTool's official documentation files for ScienceWorld and ALFWorld.

```
You are writing a precondition catalog that a runtime
checker uses to decide whether a tool call proposed WITHOUT
consulting the language model can be executed safely in a
text-based environment.  For every tool below, list the
preconditions that must hold for the call to be executable
by the environment, using ONLY predicates from the library
applied to the tool's own argument names.  Include a
precondition only if a call that violates it would be
refused or would be meaningless; when in doubt, leave it
out.  Tools with no arguments need no entries.  Return ONLY
a JSON object mapping each tool name to a list of objects
of the form {"slots":  [<argument names>], "template":
<predicate name>}; unary predicates take one argument,
distinct takes two.

Predicate library (each predicate is evaluated against the
agent's observation history and returns Satisfied, Violated,
or Unknown when the evidence is missing):
- visible(x):  the object x is listed in the latest
observation of the agent's current location.
- in_inventory(x):  the object x is currently carried by the
agent.
- reachable(x):  x is visible or carried (visible(x) or
in_inventory(x)).
- container(x):  an observation has shown that x can contain
or support other objects (e.g., "a table (on which is ...)",
"a cup (containing ...)").
- known_location(x):  x is the name of a location (room)
that has appeared in an observation.
- not_current_location(x):  x is not the location the agent
is currently in.
- distinct(x, y):  x and y refer to different objects.
```

Table 5: Mined catalog (all 6,213 archived steps; the replay in Table 3 uses the two cross-fitted fold catalogs) and LLM-written catalog. Slot names follow the trajectory parser.

Tool	Mined	LLM-written
`activate`	container(OBJ); not current location(OBJ); visible(OBJ)	reachable(OBJ)
`clean`	–	in inventory(obj); reachable(recep); distinct(obj, recep)
`close`	–	reachable(OBJ)
`cool`	–	in inventory(obj); reachable(recep); distinct(obj, recep)
`deactivate`	–	reachable(OBJ)
`dunk`	–	reachable(OBJ_to_dunk); reachable(OBJ_liquid_container); container(OBJ_liquid_container); distinct(OBJ_to_dunk, OBJ_liquid_container)
`eat`	–	reachable(OBJ)
`examine`	–	reachable(OBJ_tool)
`flush`	–	reachable(OBJ)
`focus_on`	not current location(OBJ)	reachable(OBJ)
`go`	not current location(recep)	–
`go_to`	not current location(LOC)	known location(LOC); not current location(LOC)
`heat`	–	in inventory(obj); reachable(recep); distinct(obj, recep)
`look_at`	not current location(OBJ); visible(OBJ)	reachable(OBJ)
`look_in`	not current location(OBJ_container); visible(OBJ_container)	reachable(OBJ_container); container(OBJ_container)
`mix`	not current location(OBJ_container)	reachable(OBJ)
`move`	not current location(OBJ_source); distinct(OBJ_source, OBJ_target); not current location(OBJ_target)	reachable(OBJ_source); reachable(OBJ_target); distinct(OBJ_source, OBJ_target)
`open`	not current location(OBJ)	reachable(OBJ)
`pick_up`	not current location(OBJ)	visible(OBJ)
`pour`	not current location(OBJ_from); distinct(OBJ_from, OBJ_to); not current location(OBJ_to)	reachable(OBJ_from); reachable(OBJ_to); container(OBJ_from); container(OBJ_to); distinct(OBJ_from, OBJ_to)
`put`	–	in inventory(obj); reachable(recep); container(recep); distinct(obj, recep)
`put_down`	–	in inventory(OBJ)
`read`	–	reachable(OBJ)
`take`	–	visible(obj); reachable(recep); container(recep); distinct(obj, recep)
`toggle`	–	reachable(target)
`use`	not current location(OBJ_target); visible(OBJ_target); distinct(OBJ_target, OBJ_tool); in inventory(OBJ_tool); not current location(OBJ_tool); visible(OBJ_tool)	reachable(OBJ_tool); reachable(OBJ_target); distinct(OBJ_tool, OBJ_target)

Prompt for LLM select. One call per decision point, with the environment's name, the simulator's task description, the last three actions and observations, the tool's documentation, and the gate's candidate lists ($K{=}8$ per argument) filled into the template below. A reply is valid if it names one candidate for every argument (case and surrounding whitespace ignored) or is the word ASK; no reply was invalid.

```
You fill in the arguments of the next action of a text-based
agent in <environment>.  The next tool is fixed.  Choose one
value for every argument from its candidate list, or answer
ASK if no combination of candidates is likely to be executed
correctly in the current state.
```

```
Task: <task description>
Recent steps (oldest first):
> <action>
<observation>
...
Tool: <tool> - <tool description>
Arguments and candidates:
- <argument>: [<candidate>, ...]
Answer with only a JSON object that maps every argument
to one of its candidates, for example {<argument>:
<candidate>, ...}, or with the single word ASK.
```

C COUNTERFACTUAL-EXECUTION PROTOCOL

Decision points. For every logged episode we rebuild, step by step, the candidate store from the successful calls in the last $W=2$ records (AutoTool's native window) and the evidence tracker from the observations, exactly as the gate sees them online in our closed-loop runs. A step is a decision point if its logged tool has at least two arguments and the greedy filler completes every slot. Each gate's tuple is computed at that step; the logged tool is used for all gates, which isolates argument binding from tool prediction.

Restoration and execution. A gate's tuple is rendered with AutoTool's action generator. If it equals the logged action (ignoring case) we reuse the logged observation. Otherwise a fresh simulator is reset to the same task and variation, the logged native action sequence of the prefix is replayed, and every intermediate observation and score must match the log exactly; the tuple is then executed once. Identical actions from different gates at the same point are executed once. A replay whose prefix diverges is retried once and otherwise marked undecidable. Replays from earlier runs of the same pipeline on the same logs are reused when the state key (episode, step) and the action coincide.

Refusal label. An executed action is refused if its observation contains a failure message of the simulator. We extracted the failure messages from the action handlers and device objects of the ScienceWorld engine, for example *you can't move something into itself*, *that can't be moved there*, *is not moveable*, *you can't pick up a liquid directly*, *I'm not sure how to use*, and the parser's *no known action matches that input* (38 phrases, listed in the released code), and apply them to the stored observation of every execution, including replays reused from earlier runs, and of every logged model action. To check coverage, we grouped the observations of all 902 distinct executed actions by their wording with object names masked: each of the 6 groups labeled accepted reports an effect (*you move the X to the Y*, *you pour the contents of the X into the Y*, *the thermometer measures a temperature of N degrees*), and each of the 11 groups labeled refused is a failure message. The label measures whether the environment executed the action, not whether the action advanced the task; usefulness is measured in closed loop through task completion.

ALFWorld. The development logs are one run of native AutoTool and three runs each of the reject and CRA arms under default reuse on all 134 test games (938 episodes, 24 September). A state is restored by loading a fresh AgentBoard ALFWorld environment with the single game file of the task and replaying the logged actions; replaying all 938 episodes in full matched every logged observation. A step is a decision point if its logged action has at least two arguments, which excludes navigation (*go to* names one receptacle). An action is refused if the observation is *Nothing happens.*, the message of ALFWorld's text engine for a command that does not apply. Every one of the 3,610 cached tuples that differ from the logged action, executed by the gates or by the oracle, was refused. To rule out a restoration artifact, we restored 60 randomly drawn points (52 games) and executed actions that the logs do not contain but that should apply: *inventory* was accepted at 60 of 60 points and navigation to another visited receptacle at 46 of 46. ALFWorld's preconditions are tied to the agent's location and hand, and AutoTool's ALFWorld output parser records no objects seen on arrival, so the cache mostly holds receptacle names and the objects the agent has taken, and a cached tuple applies only when it reproduces the logged action. AutoTool's ALFWorld action generator renders `heat` and `cool` with the verb *clean*; the counterfactual renders them with their own verbs to measure argument binding alone, while the closed loop keeps AutoTool's generator.

For the ALFWorld closed loop we added one predicate to the expert catalog, *the target of* `go` *is not the current location*, and scoped rules by environment, so the ScienceWorld rule for `use` no longer applies to ALFWorld's one-argument `use`; ScienceWorld verdicts are unchanged. AutoTool's tool graph declares two slots for `go` (`recep` and an optional `through`), so in closed loop the gates treat navigation as a two-slot tool and CRA repairs stale targets; the 86 refused model-free navigation actions of CRA with Gemini all target an object rather than a receptacle (*go to soapbar 3*), which no predicate checks. Each arm runs on its own core through the games in the same order. Both ALFWorld runs were interrupted once, when an external change to the shared Python environment made the tool predictor fail to load; the affected episodes had ended with that exception before any model call and were rerun under the same plan and sources, keeping the completed first slice of 23 games. One job of the Claude run was killed by the shared memory limit after one game and resumed under the same plan; the runner skips games already logged.

Held-out tasks and the irreversibility rule. The development tasks are AgentBoard's 90 ScienceWorld tasks without the three *lifespan* types and *identify-life-stages-1*. For each of the remaining twelve types, the held-out run takes as many ScienceWorld test variations as the development set has, the lowest-numbered ones that AgentBoard does not use: 327–334 (melting point of a known substance), 225–230 (melting point of an unknown substance, and each of the three *find* types), 405–412 (thermometer), 21 (freeze), 27–29 (secondary and tertiary paint), 21–24 (boil, change of state), and 24–26 (chemistry mix); the task description is the simulator's own. The rule sends a proposal to the model whenever its tool is `focus on`, `eat`, or `mix`, which AutoTool's tool documentation describes as *indicates task completion if the object is correct*, *consume an edible item*, and *chemically combine*. It acts before candidates are retrieved, in both filler arms, and leaves every other tool unchanged. Because the rule returns before the binding step, we count its hand-backs from AutoTool's own log of tool predictions and filling fallbacks. In both closed-loop runs, a task on which an arm received an empty model response was rerun once with all five arms and excluded if it failed again, as fixed before the runs: three tasks were rerun in the development run and five in the held-out run, and one held-out task (a *boil* task) was excluded.

Same-state branches. Table 7 contrasts this protocol with the other two. States were fixed before any branch ran, and so were the outcomes: the paired success difference, with the exact McNemar test for the first runs and a task-bootstrap interval over all runs, and, as secondary outcomes, steps, calls, and tokens after the branch point; the task-level sign tests and the repeat analyses of Appendix D were added afterwards. The ScienceWorld states are the 51 common decision points (from 17 tasks) at which the logged action came from the model and the action of CRA with the expert catalog was accepted and differs from it. The ALFWorld states are, for each closed-loop game of CRA, the first model-free navigation that CRA repaired and the environment accepted (96 of the 134 games with Gemini, 97 with Claude); the unrepaired proposal had failed the certificate, so the reject gate would have asked the model there. A branch restores its state by replaying the logged prefix and checking every observation (and, in ScienceWorld, the score); the agent's memory records each replayed step as *Action:* a followed by its observation, so the two branches of a state share the same prefix and differ from the branch action on. The ScienceWorld model branch executes the logged model action and the ALFWorld model branch calls the model. From then on the threshold τ is infinite, so every action comes from the LLM, while AutoTool's rule that checks the valid actions after two refusals stays in place; an episode ends when the task is solved or failed, or after 30 steps in total, as in the closed loop, and the steps of a run that fails count to its end. The endpoints are not deterministic at temperature 0: between the first two runs of the same branch, the outcome changed at 35% of the ScienceWorld states for the model branch and at 35% for the CRA branch. We therefore ran every branch three times and average success within a state before comparing branches; the McNemar test uses the first run, the analysis registered before the repeats.

Data and freeze timeline. Table 8 lists every data source, when it was produced, and what it was used for. The counterfactual study runs on development states that we also inspected while building the tracker: the location predicate was added after the failure analysis of Section 5.1, and the refusal audit of Section 5.2 exposed two tracker defects (a feedback sentence parsed as an object, and substring name matching in the template library) that we fixed before any closed-loop run. The closed loop is therefore the only evaluation on runs that did not influence the gate; all catalogs, the tracker, and the gate code were frozen, with source hashes recorded, before it started. The

Table 6: **Same-state branches.** One branch executes the accepted action of CRA, the other the model's (ScienceWorld: logged; ALFWorld: a fresh call, as the reject gate would); both continue with the same LLM without model-free actions, three runs each. n: states; *Diff.*: model minus CRA in percentage points (task-bootstrap 95% CI); p: exact McNemar, first runs; *Extra steps*: CRA's branch minus the model's; *Nav.*: runs whose model-chosen first action is a navigation.

		Success (%)					
Setting	n	model	CRA	Diff. [CI]	p	Extra steps [CI]	Nav.
ScienceWorld, Gemini	51	56.2	49.0	7.2 [0.0, 17.8]	0.092	0.9 [−0.2, 2.2]	–
ALFWorld, Gemini	96	98.6	99.3	−0.7 [−2.1, 0.0]	1.000	1.8 [1.5, 2.1]	24%
ALFWorld, Claude	97	86.6	83.2	3.4 [−0.7, 7.9]	0.289	2.0 [1.3, 2.8]	51%

Table 7: The three evaluation protocols. Each answers a different question, so their numbers are not interchangeable.

	Counterfactual execution	Same-state branches	Closed loop
States	logged development states (563 ScienceWorld, 1776 ALFWorld)	51 of those (ScienceWorld); first repaired navigation per ALFWorld game (96, 97)	every step of fresh episodes
Tool	logged (the model's at agreement points)	logged (ScienceWorld); predicted by AutoTool (ALFWorld)	predicted by AutoTool
ALFWorld navigation	excluded (logged `go to` fills only *recep*)	the only branch type	*recep* plus optional *through*; repaired
After the gate	nothing (one step)	LLM only, to the end, three runs	the gated agent, to the end
Heat/cool renderer	fixed	as logged	AutoTool's (renders *clean*)
Held out	no	no	ScienceWorld held-out run only
LLM	none (LLM select: one call)	Gemini; Claude in ALFWorld	Gemini; Claude in ALFWorld
Estimand	refusal and agreement of one action	effect of one action on the episode	effect of the gate on episodes

irreversibility rule was designed after the development closed loop, so we evaluate it only on the held-out tasks, with outcomes and comparisons registered before that run. The ALFWorld closed loop reuses the games of the ALFWorld development runs, which motivated its navigation predicate, and is not held out.

D ADDITIONAL RESULTS

Closed-loop paired differences. Tables 9 and 10 give the paired differences behind Section 5.4.

Gate frontier. Figure 4 plots the ScienceWorld columns of Table 1.

Grounding errors. Of CRA's 51 ScienceWorld refusals (Section 5.2), 46 violate preconditions the catalog does not encode and 5 come from grounding: 4 name objects the parser does not accept, mostly location-qualified names that the tracker recorded as identities (*use thermometer in inventory on ...*: *No known action matches that input*), and 1 names the agent as an object (*move agent to fridge*). Normalizing qualified names and excluding the agent from object slots would remove these; the remaining refusals are the ones Proposition 2 permits, preconditions outside the catalog.

Held-out counterfactual audit. We apply the counterfactual protocol unchanged, with the tracker and all catalogs frozen, to the 179 decision points of the held-out closed-loop logs. Of these, 86 from 20 tasks are common states; the replays of the others do not reproduce the logged prefix (for example,

Table 8: Data sources and freeze points (September 2026 unless stated).

Source	Produced	Used for
Archived AutoTool trajectories (15 files, 6,213 steps)	2025, by AutoTool's authors	mining (two cross-fitted folds and full catalog); archived replay (Table 3)
Development runs (445 episodes, default reuse)	23	failure analysis (Section 5.1); counterfactual decision points
Expert catalog	frozen 24	location predicate added after the development failure analysis
LLM-written catalog	25, one call	archived replay, counterfactual study, closed loop
Tracker and template grounding	frozen 25	fixed after the refusal audit on development states
Counterfactual execution	24–25	Section 5.2 (development states)
Closed loop, development tasks (58, five arms)	25, after all freezes	Section 5.4
Irreversibility rule	25, after that run	held-out closed loop only
Closed loop, held-out tasks (58 run, 57 analyzed; five arms)	25, after the rule	Section 5.4
ALFWorld development runs (938 episodes, default reuse)	24	stale-navigation analysis; ALFWorld decision points
ALFWorld navigation predicate, rule scoping	frozen 25	ALFWorld closed loop (no effect on its decision points)
LLM select, execution oracle	25	Section 5.2
Closed loop, ALFWorld (134 games, four arms, two LLMs)	25, after that freeze	Section 5.4
Same-state branches (ScienceWorld and ALFWorld states above)	25–26, states fixed first	Section 5.3

Table 9: Paired differences in closed loop, per episode, with 95% task-bootstrap intervals; tokens in thousands; success and failures in points. *MF ref.*: refused model-free actions. *Fail*: episodes ending in task failure; *(focus)*: ended by a model-free `focus on`. +*ask*: `focus on`, `eat`, and `mix` always go to the model. Percentile intervals can exclude zero where an exact McNemar test does not reject; with few discordant tasks we read the test.

Comparison	Succ.	Calls	Tok. (k)	MF ref.	Fail	Fail (focus)
Development tasks						
CRA − unguarded	−1.7	−0.3	−12.0	−1.88	6.9	6.9
	[−15.5, 13.8]	[−2.0, 1.3]	[−45.1, 11.3]	[−2.40, −1.38]	[−1.7, 17.2]	[0.0, 15.5]
CRA − Reject	3.4	−0.7	−5.4	−0.02	−8.6	−1.7
	[−6.9, 13.8]	[−2.8, 1.4]	[−22.1, 10.7]	[−0.31, 0.22]	[−20.7, 3.4]	[−12.1, 8.6]
CRA (LLM) − CRA	−1.7	2.6	16.8	0.14	−6.9	−8.6
	[−17.2, 12.1]	[0.5, 4.7]	[0.5, 33.4]	[−0.10, 0.41]	[−17.2, 1.7]	[−15.5, −1.7]
CRA − native AutoTool	−3.4	0.8	10.1	−1.47	1.7	3.4
	[−15.5, 8.6]	[−0.9, 2.6]	[−2.0, 22.3]	[−1.93, −1.02]	[−8.6, 12.1]	[−5.2, 12.1]
Held-out tasks						
CRA − unguarded	−12.3	0.8	18.0	−1.95	7.0	7.0
	[−29.8, 5.3]	[−1.2, 2.9]	[2.0, 35.4]	[−2.54, −1.37]	[−5.3, 19.3]	[−1.8, 17.5]
CRA+ask − CRA	5.3	0.4	−0.7	−0.21	−7.0	−10.5
	[−10.5, 21.1]	[−1.7, 2.4]	[−17.3, 15.3]	[−0.54, 0.09]	[−21.1, 7.0]	[−19.3, −3.5]
unguarded+ask − unguarded	−5.3	0.9	12.9	−0.28	−3.5	−3.5
	[−21.1, 10.5]	[−0.5, 2.4]	[2.3, 23.7]	[−0.77, 0.19]	[−14.0, 5.3]	[−8.8, 0.0]
CRA+ask − unguarded	−7.0	1.2	17.3	−2.16	0.0	−3.5
	[−21.1, 7.0]	[−0.7, 3.1]	[4.5, 29.6]	[−2.72, −1.61]	[−12.3, 12.3]	[−8.8, 0.0]
CRA+ask − native AutoTool	1.8	1.9	12.6	−1.49	−1.8	−7.0
	[−12.3, 15.8]	[−0.2, 4.0]	[−1.9, 27.1]	[−1.98, −1.02]	[−14.0, 10.5]	[−14.0, −1.8]

ScienceWorld lists some objects in varying order, which our strict check treats as divergence). On the common states, the unguarded gate is refused 82.6% of the time, distinct repair 72.4%, Unknown-as-pass 63.2%, CRA with the mined and LLM-written catalogs 63.6% and 46.2%, and CRA with the

Table 10: Paired differences in the ALFWorld closed loop, per episode, with 95% task-bootstrap intervals; tokens in thousands; success in points. *MF ref.*: refused model-free actions. Percentile intervals can exclude zero where an exact McNemar test does not reject; with few discordant tasks we read the test.

Comparison	Succ.	Calls	Tok. (k)	Steps	MF ref.
Gemini 3.8 Flash					
CRA − unguarded	−0.7	0.3	1.0	0.0	−2.60
	[−3.7, 2.2]	[−0.1, 0.6]	[0.3, 1.7]	[−0.5, 0.6]	[−2.93, −2.29]
CRA − Reject	−3.0	0.6	2.1	3.8	0.70
	[−6.0, −0.7]	[0.1, 1.0]	[0.9, 3.1]	[3.2, 4.5]	[0.59, 0.82]
CRA − native AutoTool	−2.2	0.5	1.7	3.9	0.80
	[−6.0, 0.7]	[0.0, 1.0]	[0.5, 2.8]	[3.2, 4.6]	[0.68, 0.93]
Claude Sonnet 4.6					
CRA − unguarded	−7.5	1.1	2.9	1.1	−2.99
	[−14.2, −0.7]	[0.4, 1.7]	[1.2, 4.6]	[0.1, 2.0]	[−3.39, −2.62]
CRA − Reject	−11.2	0.8	2.7	3.9	0.21
	[−17.2, −6.0]	[0.1, 1.5]	[0.8, 4.6]	[2.9, 4.9]	[−0.01, 0.43]
CRA − native AutoTool	−12.7	0.4	1.7	5.0	1.07
	[−19.4, −6.7]	[−0.5, 1.3]	[−0.7, 4.0]	[3.9, 6.1]	[0.90, 1.25]

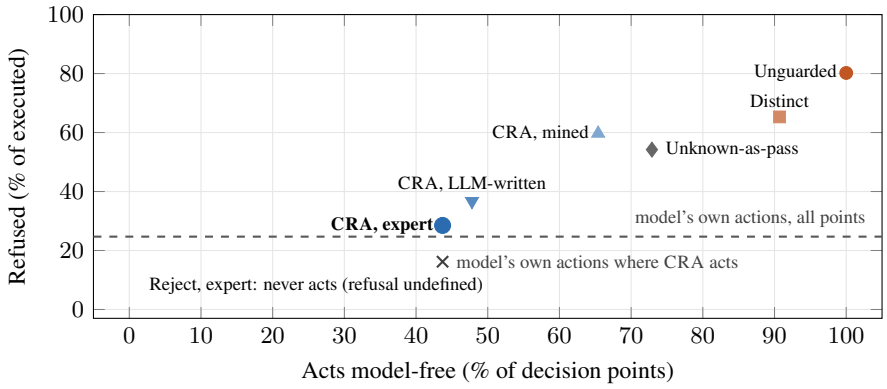


Figure 4: Gates on the 410 common decision points. x: share of points at which a gate acts model-free; y: share of those actions the environment refuses. Dashed: the model's own actions at all points; cross: the model's own actions at the points where CRA (expert) acts and the logged action came from the model (CRA is refused 33.1% of the time there). The reject gate never acts, so its refusal rate is undefined.

expert catalog 21.1% (CI [7.0, 30.0]) while acting at 44.2% of the points; the reject gate never acts, and the model's own actions are refused 16.0% of the time. Counting every undecidable outcome over all 179 points as accepted or as refused bounds CRA's refusal rate to [11.4, 34.3]%, below the lower bound of the unguarded gate ([40.8, 91.1]%). The ordering of Table 1 holds on tasks that did not shape the tracker.

Gating and repairing AutoTool's native proposals. Our gates wrap a shared slot-wise filler. At the counterfactual decision points where AutoTool itself attempted a model-free fill, the unguarded gate is refused 29.2% of the time and CRA 7.9%. To see what certification would do on AutoTool's own parameter-dependency filler, we evaluate the expert catalog, with the tracker state of that step, on every model-free action of native AutoTool in the two ScienceWorld closed-loop runs (466 actions, 193 refused). Its domain predicates certify 16 of them (3 refused), cannot certify 233 (Unknown; 154 refused, 66%), finds none Violated, and has no domain predicate for the tools of the other 217 (36 refused), such as *look around* and `focus on`. The native filler puts objects that cannot serve as tools into the tool slot of `use` (*use fridge on a counter*: *I'm not sure how to use the fridge*), and all 19 of its model-free `use` actions were refused. A reject gate on the native filler would return about half of its model-free actions to the model, two thirds of which the environment

Table 11: **Gating AutoTool's own filler in closed loop** (ScienceWorld, aggressive reuse; the 35 development tasks on which native AutoTool had executed a model-free action that the expert catalog does not certify; one run per arm). Top: per-episode means; *MF*: model-free actions (refused share). Bottom: paired differences with 95% task-bootstrap intervals, and exact McNemar tests on success (discordant tasks won:lost). The native arm is the development run of Table 2, run earlier against the same endpoint. Percentile intervals can exclude zero where an exact McNemar test does not reject; with few discordant tasks we read the test.

Arm	Succ. (%)	Calls	Steps	Tok. (k)	MF (ref.)
AutoTool, native filler	37.1	18.6	25.3	94.5	6.0 (44%)
+ Reject (expert)	17.1	19.5	23.6	105.0	3.3 (9%)
+ CRA (expert)	34.3	19.7	23.9	117.3	3.3 (10%)

Paired difference	ΔSucc. (pts)	McNemar	ΔCalls	ΔSteps	ΔMF refused
CRA − Reject	17.1 [2.9, 31.4]	7:1, p=0.070	0.2 [−2.1, 2.5]	0.3 [−2.5, 3.2]	0.1 [−0.3, 0.4]
CRA − native	−2.9 [−20.0, 14.3]	4:5, p=1.000	1.1 [−0.5, 2.8]	−1.3 [−3.4, 0.8]	−2.3 [−2.8, −1.7]
Reject − native	−20.0 [−37.1, −5.7]	1:8, p=0.039	0.9 [−1.0, 2.8]	−1.7 [−4.5, 1.2]	−2.3 [−2.9, −1.8]

refused, and leave the uncatalogued ones unchanged. The native filler rarely puts one entity into several slots (5 of its 29 multi-argument actions, against 24 for our shared filler at the same steps). Run on its proposals with the same tracker and candidates (selection offline, execution below), CRA keeps the 233 certified or uncatalogued ones, asks at 154, and repairs 79: 80% of the repaired tuples repeat an action already taken in the episode and 25% the previous one, against 29% and 0% of the native actions at the same steps. All 35 repaired navigations return to a room visited before, as in ALFWorld, and 16 of the 34 repaired `pick up` actions repeat the previous action. Executed from restored states, 39 of the 79 repaired tuples are decidable (the replays of the other 40 diverge in the prefix, also after a retry); 33 of these are accepted, including 30 of the 31 that repeat an earlier action of the episode and 13 of the 14 navigations, while none of the 5 `use` repairs is; only 3 accepted repairs repeat no earlier action, and the native actions at the same steps were refused 24 times. Counting every undecidable repair as accepted or as refused bounds the refusal rate of all 79 repairs between 8% and 58%. Replay therefore stems from recency-ranked repair, not from the slot collisions of our filler.

AutoTool's own filler in closed loop. Table 11 runs both gates on AutoTool's own proposals: a proposal executes if the expert catalog, with the tracker state of that step, certifies it or has no domain predicate for its tool; otherwise the reject gate asks the model, while CRA first searches the same candidates for a certified tuple, for single-argument tools too. The 35 tasks are the development tasks on which the native arm of Table 2 had executed a model-free action that the catalog does not certify; tasks, arms, and outcomes were registered before the run, each arm ran once, and the native arm is that earlier run. Both gates remove most refused model-free actions at about the calls of the native filler. Of the 18 repairs of CRA that the environment accepted, 14 repeat an earlier action of the episode (11 the previous one), and all 3 repaired navigations return to a room visited before. The success difference between the gates (17.1 points) is measured with little precision: every pair decoupled through the model before the gates first decided differently (next paragraph).

Where paired episodes part. Two arms of one task share the filler and the prompt, so their episodes can part only where the gates decide differently, where the model acts differently, or where the observations differ. Table 12 takes, for every pair, the first step at which the actions differ and checks whether the gates decided differently there; where both asked the model, the model is the cause. In ScienceWorld almost every pair parts where the model acted differently after identical observations, often at the first step, because the endpoint is not deterministic at temperature 0; in ALFWorld most pairs, and most of those with different outcomes, part at a gate decision. First parting diagnoses coupling, not the cause of an outcome: gates still act on decoupled episodes, and a pair that first parts at a gate decision can end differently for later reasons. In ScienceWorld one run per task thus compares the gates under largely independent model noise, which makes closed-loop differences there imprecise without excluding gate effects; the same-state branches of Section 5.3

Table 12: **Where paired closed-loop episodes first part.** For each task, the first step at which the actions of the two arms differ: at a *gate* decision (one gate asked while the other executed, or they executed different tuples), or where the *model* acted differently although both gates decided alike (in parentheses: after identical observations). *Step 1*: pairs that part at the first step. *Discordant*: tasks solved by exactly one arm (in parentheses: of these, parting at a gate decision).

Tasks (arms)	Pairs	Identical	Gate	Model (same obs.)	Step 1	Discordant (gate)
ScienceWorld						
Development (unguarded vs. CRA)	58	2	2	54 (51)	37	19 (0)
Development (reject vs. CRA)	58	0	0	58 (54)	36	10 (0)
Held-out (unguarded vs. CRA)	57	0	6	51 (51)	35	25 (3)
AutoTool's filler (reject vs. CRA)	35	0	0	35 (35)	25	8 (0)
ALFWorld, test games (reject vs. CRA*)*						
Gemini 3.8 Flash	134	12	102	20 (20)	14	4 (3)
Claude Sonnet 4.6	134	2	93	39 (39)	28	17 (12)

Table 13: **What accepted actions do** at the 376 ScienceWorld common points where the model acted (Figure 2). *Same*: equal to the model's action. Accepted actions are *new* to the episode, re-read a *measurement* (`use thermometer on ...`, which changes nothing), or re-execute a *state change* already made in the episode (`move`, `pour`, other `use`). CRA re-executes state changes in 35.4% of its accepted actions, the model in 3.9% (difference 31.5 points, task-bootstrap CI [19.2, 51.1], 34 tasks).

					Accepted: new or repeated		
Gate	Acts	Refused	Accepted	Same	New	Measurement	State change
Unguarded (G_{U})	376	325	51	0	51	0	0
Distinct repair	339	239	100	39	11	39	50
Unknown-as-pass	268	160	108	46	18	40	50
CRA (expert)	148	49	99	48	24	40	35
LLM select	83	4	79	62	32	45	2
Model (logged action)	376	93	283	283	200	72	11

restore coupling up to the branch point. The irreversible-action rule asks before any binding and records no binding event, so this analysis does not cover it.

Renderer exposure. AutoTool's ALFWorld action generator writes model-free `heat` and `cool` actions with the verb `clean`. In the closed loop only the unguarded arm executed such actions (4 steps with Gemini, 6 with Claude, all refused); native AutoTool, the reject gate, and CRA executed none. Excluding the two affected task types (44 games) leaves 90 games, on which CRA changes success relative to rejection by -2.2 points with Gemini (CI [-5.6, 0.0], McNemar 0:2, p=0.500) and -13.3 with Claude (CI [-21.1, -6.7], 0:12, p <0.001), and relative to native AutoTool by -2.2 and -13.3 points (p=0.500 and p=0.002).

Execution oracle. An execution oracle executes every cached tuple, at all ALFWorld points and at 150 randomly sampled common ScienceWorld points, to measure how often the cache contains an accepted tuple at all. It finds an accepted cached tuple at only 17.5% of the ALFWorld points and 50.3% of 149 sampled ScienceWorld points; CRA reaches 56.1% and 66.7% of these.

LLM select and CRA. Where both act in ScienceWorld they pick the same tuple in 91 of 105 cases, and CRA's refusals concentrate where the LLM declines (45% refused there vs. 17% where it acts).

Alternative ranking. To separate the recency ranking from what the cache supports, we re-select CRA's tuple at the ScienceWorld decision points with two rankings that exclude replay: *no previous action* drops certified tuples whose rendered action equals the previous action, and *no earlier action*

drops those equal to any action already taken in the episode; the highest-scoring remaining certified tuple is executed from the restored state, and CRA asks if none remains (the original ranking reproduces CRA's tuple at all 563 points). On the 410 common states at which all three are decidable, recency, no previous action, and no earlier action act at 43.7%, 37.6%, and 37.3% of the points and are refused 28.5%, 74.7%, and 84.3% of the time. Where the model acted (376 states), 99, 39, and 24 of their actions are accepted, and 48, 14, and 7 of these equal the model's action. Excluding replay thus mostly trades accepted repeats for refused alternatives, which by Proposition 2 violate preconditions outside the catalog or rest on grounding errors.

Cost with fallback. LLM select pays a selection call at every decision point (350 and 236 input tokens in ScienceWorld and ALFWorld, 6% and 18% of a full agent call) and a full agent call wherever it declines. Per decision point, the expected input of these calls is 78% and 101% of a full agent call in ScienceWorld and ALFWorld (4,638 and 1,301 tokens), against 56% and 87% for CRA and 100% and 97% for the reject gate, which pay only the full calls they fall back to; the unguarded gate pays none but executes 0.80 and 0.94 refused actions per point (CRA: 0.12 and 0.04; LLM select: 0.01 and 0.00). In ALFWorld the selection calls cost more than LLM select saves.

Search. On the 563 ScienceWorld decision points, the first certified tuple in rank order equals the highest-scoring certified tuple at 204 of the 204 points at which the expert catalog admits a repair. On the 802 multi-argument binding calls of the archived replay, measured alone on one core, the search takes 5.7 ms at the median and 12.1 ms at most and never reaches its budget (0 budget fallbacks).

Undecidable outcomes. Over all 563 ScienceWorld decision points, counting every undecidable outcome as accepted or as refused bounds the refusal rate of CRA with the expert catalog to [25.0, 35.3]%, of the LLM-written catalog to [31.9, 43.8]%, of the Unknown-as-pass gate to [44.4, 62.4]%, of distinct repair to [48.9, 73.5]%, and of the unguarded gate to [59.0, 85.6]%. The upper bound for CRA lies below the lower bound of every gate without certificates.

Ablations. Table 14 gives the counts behind Section 5.2. Container-role evidence protects reliability (treating every visible object as a container adds 21 refusals), and qualified referents (*thermometer in inventory*) protect coverage (without them, CRA accepts 33 fewer actions). *Cleaned distinct* repairs without certificates from a cleaned cache that keeps short strings naming visible or carried objects and forces distinct values; it is refused 43% of the time.

Table 14: Ablations of CRA with the expert catalog, and repairs without certificates, on the 437 decision points at which all variants are decidable.

Variant	Accepted	Refused	Ask	Refused (%)
CRA, expert	131	51	255	28
Cleaned distinct	95	73	269	43
CRA − identity	131	51	255	28
CRA − qualified referents	98	51	288	34
CRA − container role	118	72	247	38
Unknown-as-pass CRA	140	164	133	54
Distinct repair	133	246	58	65

Unique certificates. One might act model-free only where a single cached tuple is certified. At the ScienceWorld points where the model acted, CRA's executed actions that differ from the model's (accepted or refused) occur where the model's tuple is not certifiable in 90 of 98 cases, and a unique certified tuple is the model's in only 1 of 46 cases (against 49 of 102 when several are certified); in ALFWorld, 51 of CRA's 225 actions miss the model's tuple because it is not certifiable. Uniqueness is no proxy for the model's choice.

Catalog coverage in closed loop. Table 15 collects the sources of cost measured across our protocols. The expert catalog has domain predicates for `go to`, `move`, `use`, `pour`, `dunk`, and `pick up`; every other tool passes on record checks alone. On the ScienceWorld development tasks, 112

Table 15: **Where the costs of model-free actions come from**, for CRA with the expert catalog. Counterfactual: development decision points (Table 1); closed loop: aggressive setting (Table 2); branches: same-state continuations, three runs (Figure 3); †later runs. –: not measured.

Source	Evidence	ScienceWorld	ALFWorld
Refused: precondition outside the catalog	Counterfactual	46 of 51 refusals	55 of 63 refusals (agent not at the receptacle); with that predicate, refusal 26.6% → 4.4%
Refused: grounding error	Counterfactual	5 of 51 refusals	–
Accepted: replay of the recent past	Counterfactual; closed loop	35 of 99 accepted actions re-execute a state change already made (model: 11 of 283); closed loop: 31 of 36 repeat the previous action	closed loop: all 337 (Gemini) and 436 (Claude) accepted navigation repairs go back
Cost of that replay	Branches	model's branch succeeds 7.2 points more (CI [0.0, 17.8]; task sign test p=0.302)	+1.8 and +2.0 steps, +0.5 and +0.9 calls per episode; success not detectably changed
Replay replaced by asking	Closed loop†	–	success +2.2 and +9.7 points relative to CRA (Claude: 15 vs. 2 games, p=0.002), −3.2 and −2.8 steps per episode
Uncatalogued pass-through	Closed loop	112 of 171 (development) and 144 of 166 (held-out) model-free actions; model-free `focus on` ends 5 of 7 and 6 of 11 failures	–
Irreversible or uncatalogued actions sent to the model	Held-out closed loop	fatal model-free `focus on` 6 → 0 under both rules; task failures 11 → 7 (irreversible) and 10 (fail-closed†)	–

of CRA's 171 model-free actions use such uncatalogued tools (7 refused) and 59 catalogued ones (5 refused), against 107 refused of 125 catalogued actions for the unguarded filler; on the held-out tasks, 144 and 22 (19 and 4 refused). The fatal model-free `focus on` actions of Section 5.4 (5 and 6 for CRA) are all uncatalogued. Run on the held-out tasks, a fail-closed CRA that sends every uncatalogued tool to the model (registered separately and run a day later against the same endpoint) handed 72 such proposals to the model and executed none model-free; it removed the fatal model-free `focus on` endings but not the failures and cost 1.0 calls (CI [−0.8, 2.9]) and 44.2k tokens (CI [24.9, 64.0]) per episode; relative to the unguarded filler of the earlier run, it changes success by −19.3 points (6 vs. 17 tasks, p=0.035). The rule that sends only `focus on`, `eat`, and `mix` cost 0.4 extra calls per episode. Certification thus removes the refusals where the catalog has predicates, while the fatal actions come from where it has none; the accepted but unhelpful repairs of Section 5.3 come from where it has them.

Effect-aware abstention in closed loop. An effect-aware CRA executes a model-free action only if it does not repeat an action already executed and accepted in the episode, observations (*examine*, *look*, *inventory*, *check valid actions*) excepted; otherwise it asks the model. The rule compares the rendered action with the episode's accepted actions and changes nothing else: certification, repair, and the fallback stay those of CRA. We registered it with its outcomes before running it once on all 134 ALFWorld games with each model, a day after the other arms, against the same endpoint. It asked the model 211 times with Gemini and 323 times with Claude (in 94 and 82 games), 208 and 319 of them instead of a navigation. Relative to CRA, it changes success by 2.2 points with Gemini (CI [−0.7, 6.0]; 4 vs. 1 games, p=0.375) and 9.7 with Claude (CI [4.5, 15.7]; 15 vs. 2, p=0.002), calls by −0.7 and −0.2 per episode, and steps by −3.2 and −2.8; relative to rejection, success changes by −0.7 (0 vs. 1, p=1.000) and −1.5 points (3 vs. 5, p=0.727) and calls by −0.1 and 0.6.

Repeats in closed loop. The mechanism of Section 5.3 also shows in the closed-loop runs. In ScienceWorld, 31 of the 36 multi-argument model-free actions of CRA that the environment accepted on the development tasks, and 10 of 15 on the held-out tasks, repeat the action just executed, against 17 of 97 and 3 of 77 of the model's own accepted multi-argument actions. In ALFWorld, all 337 (Gemini) and 436 (Claude) navigations that CRA repaired and the environment accepted return to the previous location, against 8% and 5% of the model's own navigations in the reject arm.

Where the branch difference concentrates. Among the ScienceWorld branch states, where the action of CRA repeats an action already taken in the episode (34 states), the branches succeed in 58.8% (model) and 52.9% (CRA) of the runs; at the other 17, in 51.0% and 41.2%. A rule that asks the model wherever CRA would repeat the previous action (27 states) would reach 52.9%, and asking at as many randomly chosen states 52.8% in expectation (difference 0.1 points, task-bootstrap CI [−5.0, 4.8]). Both rates are computed from the branch runs above, without new runs. Among all decision points where the model acted, the rule would also ask at the 34 points where CRA's repeat equals the model's action (33 of them thermometer re-readings), paying a call without changing the action; restricted to repeated state changes, it would ask only where CRA re-executes a `move`. These analyses were not registered and reuse the same runs, so they are exploratory. Neither localizes the difference in repeats: repeats describe how repair departs from the model, not which departures lower success.

Window size. In ScienceWorld, with a window of $W=4$ records instead of AutoTool's native two, the greedy proposal still fails the expert catalog at all 654 decision points. On the 447 common points, repair has more candidates to choose from: CRA acts at 62.2% of them and accepts 165 actions, but it is also refused more often (40.6%): the extra candidates satisfy every catalogued precondition yet often violate uncatalogued ones, which account for 107 of its 114 refusals over all 654 points. The ordering of the gates is unchanged: the unguarded gate is refused 81.7% of the time, distinct repair 68.2%, the Unknown-as-pass gate 59.0%, and CRA with the LLM-written catalog 49.3%. A larger window does not bring repair closer to the model: at the 413 decidable points where the model acted, 79 of CRA's 136 accepted actions differ from the model's (51 of 99 with $W=2$), 28 of them repeats of the previous action, and 62 of the 136 repeat it overall (the model: 37 of 309).

Per-tool outcomes. Tables 16 and 17 split the common decision points by tool. In ALFWorld, CRA with the expert catalog executes 56 `take` actions, 55 of which name a receptacle other than the agent's location; all 55 are refused. They reuse an object that the agent put there earlier (*take soapbar 4 from garbagecan 1* while at the countertop), so *object at the receptacle* is certified, but the catalog does not require the agent to be there. LLM select almost never proposes `take` and asks instead. *Adding the missing predicate.* We add *the agent is at the receptacle* (Unknown before the first navigation) to the expert catalog and recompute CRA on the executed tuples of these points (no new execution; the unmodified catalog reproduces CRA's choice at all 1776 points). All 55 refused `take` actions become asks, and no accepted action changes: CRA acts at 10.2% of the points instead of 13.3% and is refused 4.4% of the time (CI [1.4, 8.4]) instead of 26.6%. Requiring the same for `put`, `clean`, `heat`, and `cool` changes nothing further. The omission thus explains CRA's remaining ALFWorld refusals, but not its closed-loop cost: every accepted navigation repair returns to a known receptacle other than the current one, which both *known receptacle* and *not the current location* certify.

Calibration of certificates. Table 18 pools every distinct executed ScienceWorld tuple across gates and reports, for each catalog, the refusal rate by the catalog's verdict on that tuple. Tuples certified by the expert and the LLM-written catalogs are refused 33.0% and 39.7% of the time, against 83.5% and 84.4% for tuples they reject, whereas the mined catalog's certified tuples are refused 61.2% of the time.

Table 16: ScienceWorld counterfactual outcomes by tool on the common decision points: decision points at which the gate acts (of all points for that tool) and the refusal rate of those actions.

	`move`		`use`		`pour`	
	acts	refused	acts	refused	acts	refused
Unguarded	205/205	100%	184/184	56%	21/21	100%
Distinct repair	189/205	73%	165/184	59%	18/21	44%
Unknown-as-pass CRA	173/205	70%	115/184	34%	11/21	18%
CRA, expert	71/205	27%	108/184	30%	0/21	–
CRA, mined	178/205	71%	72/184	36%	18/21	44%
CRA, LLM-written	79/205	51%	108/184	30%	9/21	0%

Table 17: ALFWorld counterfactual outcomes by tool on all 1776 decision points, as in Table 16.

	`take`		`put`		`clean`, `heat`, `cool`	
	acts	refused	acts	refused	acts	refused
Unguarded	980/980	100%	500/500	93%	296/296	75%
Reject (expert)	1/980	0%	44/500	18%	0/296	–
Distinct repair	366/980	100%	187/500	81%	140/296	46%
Unknown-as-pass CRA	980/980	100%	500/500	93%	293/296	74%
CRA, expert	56/980	98%	181/500	4%	0/296	–
CRA, mined	980/980	100%	500/500	93%	296/296	75%
CRA, LLM-written	0/980	–	0/500	–	0/296	–
LLM select	1/980	0%	173/500	0%	134/296	0%

Table 18: Refusal rate of executed tuples by catalog verdict (all decidable executions, each distinct action counted once per decision point). The template library returns Violated rather than Unknown when a name is absent from the evidence, so the LLM-written catalog's verdicts on these tuples were never Unknown.

	Satisfied		Unknown		Violated	
Catalog	n	refused	n	refused	n	refused
Expert	200	33%	180	85%	522	84%
Mined	338	61%	38	16%	526	84%
LLM-written	237	40%	0	–	665	84%