AGENTCLUTCH: A CONTROLLED STUDY OF VERIFIER-GATED FALLBACK FOR STATEFUL AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

An agent cascade can contain a test-passing fallback yet return an earlier failing output. We study this failure with AgentClutch, a fixed-order cascade that restores a registered execution state before fallback. Our analysis separates the success available among retained outputs, the success delivered by the stopping rule, and the executor cost of acquiring it. On 134 Python repair problems, replacing syntax stopping with public-example checks, while holding candidate and reference outputs fixed, raises delivered test success from 78.4% to 90.3%. The stronger gate recovers 16 blocked test-passing fallbacks at 29.1% higher executor-token use. In CooperBench and ScienceWorld, a middle executor adds unique operational acceptances but costs more than the downstream reference work its successful stops avoid. The full cascade nevertheless remains cheaper than standalone reference execution in the CooperBench census under a fixed token tariff. These findings show why complementary outputs, reliable selection, and economical acquisition must be evaluated separately. The studies concern restorable execution units rather than uninterrupted task completion.

1 INTRODUCTION

Giving an agent another opportunity to act does not ensure that it returns a better output. A coding agent may accept an edit that passes syntax checks and stop before a later executor produces a test-passing repair. Figure 1 shows where fallback intervenes. In its upper panel, an upstream agent registers an execution unit; the checker returns an accepted state or restores the start before the next executor tries. If every executor is rejected, the unit returns failure. The lower panel holds a candidate output C_1 and reference output R_1 fixed: syntax stopping returns the failing C_1, whereas public-example stopping reaches the passing R_1. Producing a successful alternative and delivering it are separate events.

The stopping rule also determines which attempts are acquired. Each reached attempt incurs cost, while downstream work is avoided only when an earlier executor stops the cascade. An aggregate success–cost point cannot separate missing alternatives from blocked ones, and standalone acceptance cannot price an executor's insertion on the units that reach it. We measure these quantities with a fixed-order policy on restorable execution units.

Matched comparison requires restoration because a rejected attempt can mutate the workspace. We study units with fixed instructions, external postconditions, and restorable starts; the postconditions check required file and symbol changes or simulator milestones. AgentClutch tries executors in a fixed order and ends at reference executor R. ReplayPlane records matched attempts under this contract. Restoration controls inherited state but does not make stochastic reruns identical. The comparisons concern restorable units rather than uninterrupted task completion; the same contract can apply to other checkpointable tool use with reliable checks.

Our controlled repair study holds candidate and reference outputs fixed while varying only the stopping gate, recovering 16 test-passing fallbacks blocked by syntax stopping. The repository and simulator studies then examine the cost of acquiring complementary accepted outcomes. Together, these comparisons separate selection from generation and marginal insertion cost from complete-policy economics.

Our contributions are:

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.

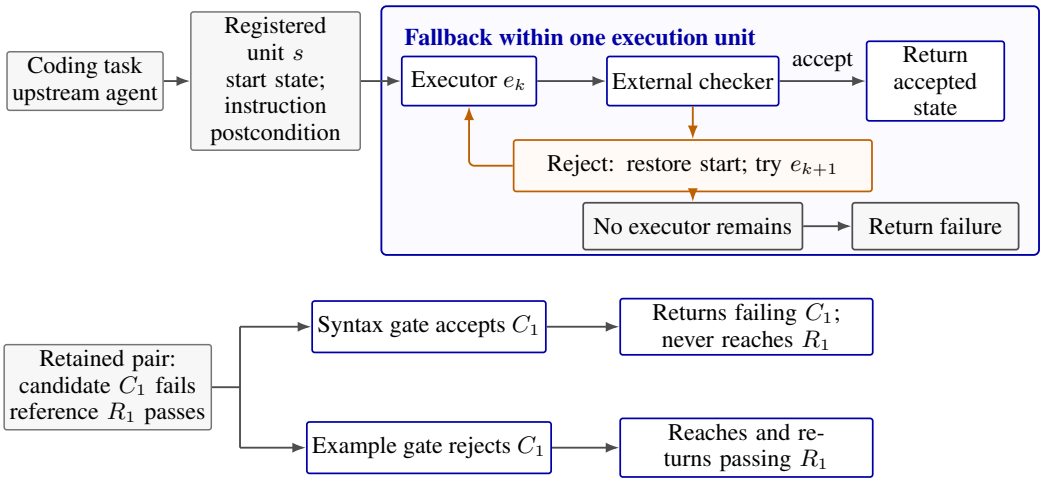


Figure 1: **AgentClutch on one restorable execution unit.** Executors run in a fixed order from the registered starting state; a rejected attempt is rolled back before fallback, and the first gate-accepted state is returned. If every executor, including the terminal reference, is rejected, the unit returns failure. The secondary panel holds a candidate output C_1 and reference output R_1 fixed under two gates: syntax stopping returns C_1 despite failed evaluation; public-example stopping rejects it and reaches the passing R_1.

1. **Selection with fixed outputs.** A controlled repair study isolates the stopping gate's effect on delivered test success without changing the candidate or reference generations, recovering 16 test-passing fallbacks blocked by syntax stopping.
2. **Conditional acquisition cost.** A cost decomposition measures the cost of an inserted executor on the states that reach it, including the downstream reference work its accepted outputs avoid, alongside its unique accepted outcomes.
3. **Cross-setting evidence and boundaries.** CooperBench and ScienceWorld show that complementary operational outcomes can carry an insertion premium, while distinguishing recorded local decisions from live, whole-task performance.

2 Related Work

Routing and cascades. Model routing and cascades ask whether another computation's expected benefit justifies its cost (Russell & Wefald, 1991; Horvitz, 1987). RouteLLM and RouterBench select across models (Ong et al., 2025; Hu et al., 2024); agent routers condition on task features, workflow progress, or trajectory evidence (Paul & Nandy, 2026; Li et al., 2026; Lei & Huang, 2026). FrugalGPT scores responses, AutoMix uses self-verification, and cascade-routing theory studies quality-estimator-based allocation (Chen et al., 2024; Aggarwal et al., 2024; Dekoninck et al., 2025). Gatekeeper tunes deferral under a near-dominance assumption (Rabanser et al., 2025). These works introduce cost-aware cascading; our study instead diagnoses selection loss by holding retained alternatives fixed and measuring which passing outputs the stopping rule delivers.

Repeated sampling and behavioral checks. Repeated sampling can improve success under automatic verification (Brown et al., 2024; Snell et al., 2025). CodeT uses generated tests and execution agreement (Chen et al., 2023); Reflexion carries verbal feedback across attempts (Shinn et al., 2023). ImpossibleBench exposes specification-violating shortcuts through conflicting specifications and tests (Zhong et al., 2026). Sampling changes the available output set; our repair intervention instead holds outputs fixed and changes only the stopping gate, so it isolates selection from generation.

Execution feedback and stateful workflows. Planner–executor systems separate decomposition from action (Shen et al., 2023; Erdogan et al., 2025); LATS searches reasoning and action branches

with environment feedback (Zhou et al., 2024). EcoAssistant combines code-execution feedback with weak-to-strong escalation (Zhang et al., 2023), and TRIM routes critical reasoning steps using process rewards (Kapoor et al., 2026). Shepherd supplies reversible execution traces (Yu et al., 2026). Our contract restores the registered start after rejection so matched fallback does not inherit earlier mutations; restoration provides matched starts for the comparison, not an established general recovery benefit.

3 AGENTCLUTCH: EXECUTION AND MEASUREMENT

3.1 PROBLEM SETTING AND EXECUTION CONTRACT

A scored unit s specifies a registered starting state, an instruction, permitted tools, a write scope, and an operational postcondition. An executor e is a complete inference configuration, including its model, reasoning mode, and budget. An *executor attempt* is a bounded run from that start; it may contain multiple model calls and tool actions. We associate each executor–unit record with an outcome and measured resource cost $c_e(s)$, including any within-executor retries permitted by the study protocol (Section 4.1).

After each executor attempt, the checker returns a passing state or rejects it and restores the registered start before the next attempt. The step instruction and start remain fixed across alternatives. ReplayPlane records each attempt's executor, resulting state, checker decision, and resource use.

3.2 AVAILABILITY, DELIVERY, AND ACQUISITION COST

AgentClutch tries an ordered portfolio, restores the registered start after failure, and ends at reference executor R. Here "reference" names the terminal role in the policy, not a claim that its standalone outcomes are always stronger. A verified pass stops execution, so even a fixed order allocates different numbers of attempts to different steps. Section 4.1 specifies the frozen portfolio; the mechanism does not prescribe a universal order.

The three quantities that organize the studies are the success available among retained outputs, the success delivered by the stopping rule, and the executor cost of acquiring a delivered output. Let R denote the reference executor and $y_e(s) \in \{0, 1\}$ be executor e's verified outcome on unit s in a complete matched matrix. Let $y_\sigma(s)$ be the policy's verified outcome on s. For any policy σ, mean acceptance is $Q_\sigma = \sum_s w_s\, y_\sigma(s)$, where $w_s \geq 0$ sum to one and are fixed across policies. Pooled analyses use uniform unit weights; task-level paired analyses are identified where reported.

Availability and delivered success. An *operational gate* uses policy-visible information to accept an output or defer. A separate *behavioral evaluation* scores the output without supplying feedback to that policy. Let $g_e \in \{0, 1\}$ be the gate decision and $B_e \in \{0, 1\}$ the evaluation-test outcome for retained output e. *Delivered test success* means returning an output with $B_e = 1$; rejecting all attempts counts as failure. For two-stage repair, let C denote the retained candidate output and R the retained reference output. Then $S_{C \to R} = g_C B_C + (1 - g_C) g_R B_R$, while raw-reference success is B_R. For a retained pair, $U = \max(B_C, B_R)$ indicates an available success. For any selection policy σ on this pair, S_σ denotes its delivered test success.

A policy can fail because none of its retained outputs passes evaluation, or because it does not return a passing output that is already available. Selection has the descriptive accounting identity

$$1 - \mathbb{E}[S_\sigma] = \underbrace{1 - \mathbb{E}[U]}_{\text{no passing output available}} + \underbrace{\mathbb{E}[U - S_\sigma]}_{\text{available success not delivered}}. \tag{1}$$

The observed union is a diagnostic bound over retained outputs, not an executable policy or a guarantee under fresh generation. Table 1 instantiates this accounting; Appendix A.1 gives identities for comparisons against the raw reference and the gate-filtered reference. A false acceptance loses an available reference success only when the paired reference also passes evaluation.

Recorded-outcome accounting. Recorded operational acceptance measures success under the policy's verifier. We separately measure behavioral validity where independent evaluation outcomes

are available. For a policy with verified stopping and terminal reference R,

$$y_\sigma(s) = y_R(s) + (1 - y_R(s))\,\mathbf{1}\{\text{an earlier candidate passes on } s\}. \tag{2}$$

On the recorded matrix, policy acceptance is the union of the executors' gate acceptances, so terminal-reference coverage follows by construction. This operational result does not establish correctness beyond the verifier's criterion. The empirical questions are which additional units are accepted and what acquiring those outcomes costs. Let K_σ denote executor cost under policy σ on a scored unit; K_σ^{tot} and $\bar{K}_\sigma$ denote its population total and mean. For candidate e before R, with attempt costs c_e and c_R, these questions separate exactly:

$$Q_{e\to R} - Q_R = \mathbb{E}[y_e(1 - y_R)], \tag{3}$$

$$\mathbb{E}[K_{e\to R}] - \mathbb{E}[c_R] = \mathbb{E}[c_e] - \mathbb{E}[y_e c_R]. \tag{4}$$

The candidate adds acceptance when it passes a reference failure; it saves executor cost when its attempts cost less than the reference computation avoided on its accepted cases. Neither property implies the other.

3.3 Acquisition cost and live execution

The cost of a cascade depends on how often each executor is reached and what its execution costs on those units. For an ordered sequence, expected cost decomposes as

$$\mathbb{E}[K_\sigma] = \sum_k \Pr(\text{reach } k)\,\mathbb{E}[c_k \mid \text{reach } k],$$

where c_k is the cost of attempting rung k. A lower-cost first attempt reduces cost when it avoids enough later computation to offset its own cost. Since token use varies by step, the relevant fallback cost is conditional on reaching that rung. For a candidate k reached with positive probability, insertion before suffix V saves cost when

$$\mathbb{E}[c_k \mid \text{reach } k] < \Pr(y_k = 1 \mid \text{reach } k)\,\mathbb{E}[K_V \mid \text{reach } k, y_k = 1].$$

Here K_V is the suffix cost avoided when k passes. Conditioning on that pass matters: replacing the last term by the cost after k fails is an unevaluated plug-in approximation. We use the exact insertion accounting in Appendix C.4. The matched matrix records reference attempts even where a reconstructed policy stops earlier, supplying the avoided reference costs. These executor costs exclude planning, checking, and restoration. A full-system objective must include those costs in compatible units; Appendix E.3 reports the available overhead sensitivity.

Equation 2 concerns replay over fixed outcomes. A fresh stochastic attempt can differ from its recorded outcome. Expected live reference coverage follows if restored reference outcomes are exchangeable with fresh reference attempts, conditional on the step and preceding failures. Semantic correctness also requires a sound acceptance contract. Restoration controls the starting state, not stochastic exchangeability. We test this boundary with fresh reference attempts in §4.6 and Appendix D.2.

4 Experiments

We ask what a fallback portfolio contains, which output its gate selects, and what it spends to obtain that output. Registered comparisons follow the protocols in Appendix E.1; additional post-sealing analyses are identified below.

4.1 Evaluation settings and endpoints

The three studies evaluate different units and endpoints, so their success notions are not interchangeable (Appendix Table 8). The controlled repair study isolates gate-induced selection loss; the CooperBench census measures complementary acceptance and executor cost; ScienceWorld examines acquisition cost in another environment.

Two conventions recur across the studies. *Execution* and *replay* reconstruct stopping decisions from separately acquired records without carrying one executor's messages into another's budget. *Cost*

charges executor tokens under a fixed tariff (CooperBench and ScienceWorld) or a token ratio (repair), always relative to a population-wide standalone reference. The CooperBench census is a prospectively collected, sealed matrix whose policy outcomes are reconstructed by replay; that differs from a policy run live as a cascade (SW4, §4.4).

CooperBench (Khatua et al., 2026) measures structural-check acceptance on 357 restored repository steps from 253 feature pairs; pair IDs are disjoint from development, although $120/165$ census features, $24/25$ base tasks, and all 12 repositories recur. ScienceWorld (Wang et al., 2022) scores native milestones after replayed gold action prefixes. Inference is paired by repair problem, clustered by CooperBench feature pair (the 253 scored pairs), and clustered by ScienceWorld episode; Appendix A.6 names each resampling unit and its base-task and repository sensitivities.

Each executor–unit record includes the complete execution assigned to it. CooperBench charges both bounded runs in a two-run arm–step cell. ScienceWorld restores each milestone interval using registered score and ordered subgoal state, without requiring byte-identical observations. Repair starts each retained edit from the original buggy code. These restoration criteria support comparisons on matched starts; composition through later model-produced states is separate. Appendices A.4 and A detail the retry contract and replay criteria.

The fixed three-stage policy (the "frozen ladder") is 14B-off $\rightarrow$ 8B-on $\rightarrow R$, with $R =$ 14B-on. Here 8B and 14B denote Qwen3-8B and Qwen3-14B; off and on indicate disabled and enabled thinking. The original CooperBench draw used an eligibility floor measured on that draw; draw 1 and the extension are development-era comparisons. The ladder was subsequently frozen for the prospective census and SW3. SW1 is a recorded transport study, and SW4 tests a two-stage policy prospectively (Appendices B.3 and E.1).

CooperBench and ScienceWorld costs apply a fixed tariff to recorded input and output tokens; repair instead weights those tokens equally. Policy totals charge all execution assigned to records the policy reaches and use the corresponding standalone-reference total as denominator. Planning, checking, and restoration overhead are excluded where unmeasured; Appendix E.3 reports the available sensitivity analysis.

4.2 Gate choice determines delivered repair success

We first isolate selection from generation. Using the same retained candidate and reference outputs, we change only the stopping gate and measure which test-passing outputs the resulting policy delivers. Of the 164 problems in HumanEvalPack Python repair (Muennighoff et al., 2024), 158 passed canonical, buggy-code, and callable-example eligibility controls before model calls; a seeded split assigned 24 to development and 134 to evaluation. Each evaluation problem is one paired unit with four retained outputs: C_1, C_2, C_3 (Qwen3-14B, thinking off) and R_1 (thinking on), each starting from the original buggy code. We reconstruct selection using a syntax gate (changed code, parseable tree, expected entry point) or that gate plus benchmark-supplied public examples. Evaluation-only tests are withheld from prompts and tools. A second, container-isolated evaluator reproduces all 536 evaluation labels and all 536 public-gate decisions (Appendix A.1).

Holding the C_1/R_1 outputs fixed, public-example gating raises delivered test success from 78.4% to 90.3%, a paired gain of 11.9 percentage points (95% interval $[6.7, 17.9]$). It uses 29.1% more executor tokens than syntax stopping. The example gate delivers all 121 available test-passing outputs in this pair; the gain measures selection rather than a change in generation (Table 1).

Table 1: **Gate choice changes delivery from the same output pair.** Both policies select from the same retained C_1/R_1 pair on 134 problems. A passing output is available on 121; neither passes on 13. Candidate false accepts count C_1 outputs accepted by the gate but rejected by evaluation tests.

Quantity	Syntax gate	Public-example gate
Delivered test-passing outputs	105	121
Available successes not delivered	16	0
Candidate false accepts	28	2
Reference attempts reached	4	30
Acquired executor tokens	439,960	568,046

The paired outputs contain 29 candidate-only and 19 reference-only test-passing cases (Appendix Table 6). Of the 134 retained pairs, 73 pass under both configurations, 29 only under the candidate, 19 only under the reference, and 13 under neither; the 121 available successes and 13 problems with no passing output follow directly. Syntax stopping accepts 28 candidate outputs that fail evaluation, but only 16 of these decisions block a passing reference. Public-example checks reject all 16, while their two remaining false accepts occur on problems where the reference also fails. A false acceptance therefore matters for recoverable success only when the blocked alternative would have passed; the 28 cell-level false accepts do not equal 28 lost successes. The example gate reaches 26 additional reference records and spends 128,086 more executor tokens to recover 16 passing outputs, about 8,005 tokens each before checker runtime.

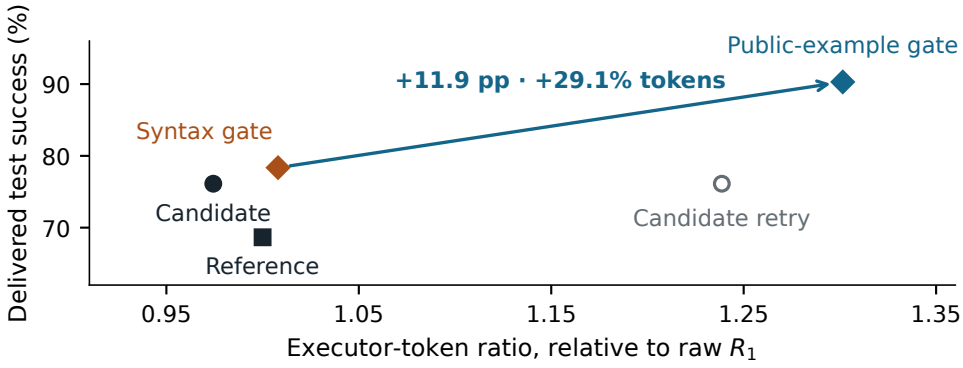


Figure 2: **Gate choice changes delivered success for a fixed output pair.** Candidate is C_1; Reference is R_1. The arrow connects the same retained outputs under two gates, not an interpolated frontier. The paired gate contrast is $+11.9$ percentage points (95% CI $[6.7, 17.9]$); the 29.1% token increase is relative to syntax stopping, not raw reference execution. Both gate points use $C_1 \to R_1$; the retry control uses $C_1 \to C_2$. Gray marks the low-diversity retry control, whose low diversity restricts interpretation rather than serving as a diverse-generation comparison.

Raw C_1 passes $102/134$ problems and raw R_1 passes $92/134$. The reference leaves the buggy source unchanged on $36/134$ problems, combining repair ability with edit omission and interface termination; Appendix A.1 gives the breakdown. The raw-reference comparison describes this configuration rather than a general ranking of reasoning modes. Figure 2 places the gate policies and retry control on one cost–success map. On 13 problems neither retained output passes.

A post-hoc sensitivity excluding Python/119, whose public-example asset exposes implementation text, yields 90.2% versus 78.9% test success, preserving the gate effect (Appendix A.1).

Retry control. The attempts-matched candidate retry is low-diversity: C_1/C_2 source is byte-identical on $124/134$ problems, and receipts cannot establish server-effective sampling settings or exclude response reuse. It describes this observed retry regime; diverse or token-matched sampling remains a separate comparison (Appendix A.1).

4.3 Most additional structural accepts come from two-stage fallback

With terminal R, nondecreasing structural-check acceptance is guaranteed in recorded replay. On the census, the frozen ladder accepts 92.7% of steps versus 81.5% for R, at 63.7% of reference price (Table 2; Appendix Figure 4a). The empirical quantities are the 40 additional structural accepts and their acquisition cost.

The 14B-off $\to R$ policy recovers 33 of those 40 additional steps, reaching 90.8% acceptance at 58.1% of reference price. Most additional accepts are thus recovered by two stages; what the remaining seven cost motivates the insertion analysis in Section 4.4. This two-stage comparison is descriptive; Appendix B.1 reports its paired estimates and intervals.

Table 2: **CooperBench census: structural acceptance and executor cost.** All 357 steps; standalone R =14B-on is the cost denominator. Bold marks the frozen ladder; † marks post-sealing policies. The secondary block lists learned and post-hoc alternatives; the essential four policies appear above it.

Policy	Accepts (of 357)	Cost ratio [95% CI]	Reference records reached
R (14B-on)	291 (81.5%)	1.000	357
14B-off (standalone)	284 (79.6%)	0.372 $[0.311, 0.445]$	0
14B-off $\rightarrow R$	324 (90.8%)	0.581 $[0.485, 0.690]$	73
Frozen ladder (14B-**off** $\rightarrow$ 8B-**on** $\rightarrow R$)	**331 (92.7%)**	**0.637** $[\mathbf{0.519}, \mathbf{0.770}]$	**42**
Secondary: learned and post-hoc alternatives			
Cost Mode (learned + fallback)	326 (91.3%)	0.603 $[0.500, 0.723]$	N/R
Static alternative† (14B-off $\rightarrow$ 8B-off $\rightarrow R$)	331 (92.7%)	0.608 (N/R)	52

The post-hoc static alternative in Table 2 also accepts $331/357$ census steps, matching the frozen ladder's total, at 0.608 rather than 0.637 of reference price. Only 326 accepted steps are shared; each policy uniquely accepts five. Equal aggregate acceptance thus conceals different covered sets. This descriptive substitution has no new interval or behavioral equivalence result (Appendix C.4).

These repository findings concern structural acceptance. Two earlier CooperBench semantic audits were withdrawn after implementation defects, and census workspaces were not retained for behavioral rescoring. A separate AI-assisted 20-state reference-phase review flagged local defects in every sampled change (Appendix D.5). The gate sweep measures agreement with the registered checker, not semantic correctness (Appendix D.1).

4.4 Complementary middle executors need not save cost

To measure the middle executor's contribution, we compare the frozen ladder with the policy that omits only 8B-on and keeps 14B-off followed by R. In the CooperBench census, 73 steps reach the middle executor. It accepts 31: 24 also accepted by the reference and seven not accepted by the reference. All 31 stops avoid reference execution, but only seven add coverage. The incurred middle-executor cost is nevertheless larger than the avoided reference cost (Table 3). In SW1 the same holds for 54 reached steps and six unique milestones, and the premium more than exhausts the saving from the first executor. In CooperBench, the complete ladder remains cheaper than standalone reference execution despite this insertion premium. The result distinguishes the cost of adding a rung from the economics of the complete policy.

Table 3: **Middle-executor insertion on reached states.** The comparison inserts M = 8B-on into 14B-off $\rightarrow R$, not the separate 8B-off $\rightarrow R$ policy. Unique accepts are M passes rejected by R; reference cost avoided is credited on all M passes. Cost entries use population-wide standalone-R expenditure as the denominator. Net intervals are post-sealing paired-cluster percentile estimates. The full partition is in Appendix Table 21. CB abbreviates CooperBench.

Population	Reached	Added accepts	Middle cost incurred	Reference cost [95% CI]	
CB census	73/357	7	0.149	0.093	$+0.056$ $[0.004, 0.112]$
SW1	54/145	6	0.862	0.270	$+0.592$ $[0.258, 1.034]$

Across five populations, the frozen ladder has higher recorded acceptance than standalone R, while its full executor-cost ratio varies (Figure 3; Appendix B). Price intervals lie below one in draw 1 and the census, cross one in the extension and SW3, and begin just above one at 1.003 in SW1.

The SW1 ladder accepts 88.3% of milestones, equal to the observed four-configuration acceptance union, at 46.8% more executor cost than R. The SW1 cost ordering is conditional on a brittle shared command interface: 61.3%–80.6% of primitive commands are unparseable across the original SW1 arms, and a 216-attempt normalization intervention changed few commands. The premium under an effectively repaired interface remains unresolved (Appendix Table 15).

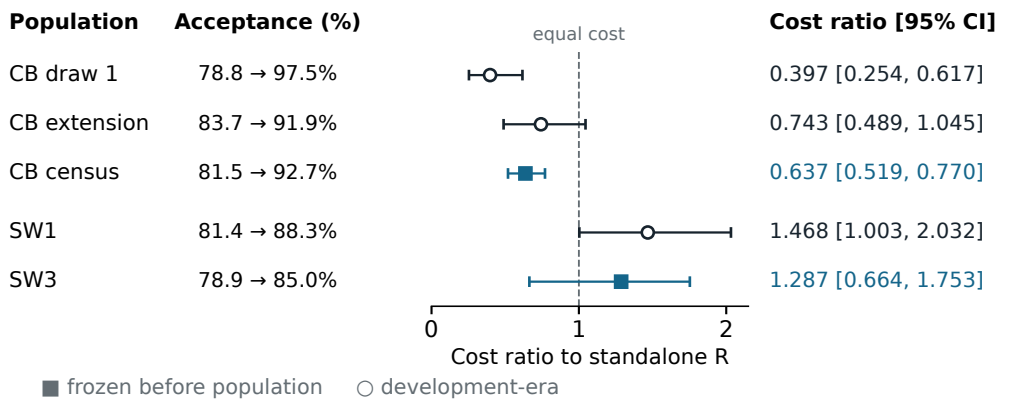


Figure 3: **Full-ladder acceptance and executor cost.** CooperBench rows (draw 1, extension, census: $80/86/357$ steps) are grouped above ScienceWorld rows (SW1, SW3: $145/147$ milestones), whose acceptance percentages are native-milestone rates and are not interchangeable with CooperBench structural-check rates. Bars are 95% task- or episode-clustered intervals (census percentile; others BCa). Squares mark policies frozen before the population was scored; circles mark development-era policies. The full-ladder points are reconstructed from recorded attempts, not run live.

A separate two-stage policy starts with 8B-off rather than 14B-off and has lower recorded cost than the full ladder in SW1 and SW3 (Appendix B). It was tested live and prospectively in SW4 on 149 milestones from 60 episodes. Its cost ratio was 0.744 with a 95% interval of $[0.435, 1.046]$, leaving cost superiority unresolved; its acceptance difference satisfied the registered point-estimate margin, not an interval-based non-inferiority test (Appendix B.5). SW4 is therefore reported separately from the reconstructed full-ladder points in Figure 3: it evaluates a different two-stage policy live rather than replaying the frozen ladder.

4.5 Alternative compute-allocation controls

The controls assess whether the observed benefits depend on switching executors, acquiring additional attempts, or selecting a configuration before execution. In repair, reference fallback returns more test-passing outputs than the observed candidate retry at matched attempt limits, with the interpretation restricted to the low-diversity regime in §4.2. On CooperBench draw 1, the ladder uses less executor cost than reference retries, while its joint acceptance–cost advantage over candidate retries remains unresolved (Appendix C.2). These comparisons characterize the tested policies rather than establishing a general ranking of switching and resampling.

The contextual-routing comparison asks what changes when a one-shot selection policy is extended with verified recovery. On the CooperBench census, acceptance rises from 79.3% to 91.3%, with additional execution and changes to stopping and restoration. This comparison concerns complete policies rather than the isolated information supplied by feedback. The learned recovery policy also has no established joint acceptance–cost advantage over the frozen ladder (Table 2; Appendices B.1 and C.6).

4.6 Discussion and design implications

Gate evaluation must consider blocked alternatives. For fixed outputs, permuting a portfolio preserves its accepted union even when some gate acceptances fail independent evaluation; what changes is the selected output. An early false acceptance prevents a later test-passing alternative from being returned only when that blocked alternative would pass. Cell-level false acceptance is therefore not identical to lost deliverable success, and aggregate checker agreement captures neither consequence.

Executor inclusion must use conditional costs. A middle executor's cost is paid on every unit that reaches it, while downstream reference work is avoided only when it accepts. It can therefore add

unique accepted outcomes and still raise total expenditure. Standalone quality and average cost do not determine its value; the relevant quantities are reach, acceptance conditional on being reached, and the reference work its successful stops avoid (Appendix C.4).

Recorded local comparisons need prospective validation. Restoration gives every executor the same registered start, but it does not make stochastic reruns interchangeable or local decisions compose across a full task. The fresh-reference comparison remains inconclusive at registered precision (Appendix D.2), and the restoration ablation leaves its direct success benefit unresolved (Appendix D.3). Matched starting conditions and repeatable outcomes are separate requirements for a live cascade.

5 Limitations and future directions

The three studies support different kinds of claims: repair isolates selection loss under fixed outputs, CooperBench and ScienceWorld measure operational acceptance and acquisition cost on restored units, and none establishes whole-task behavioral success. The boundaries below follow from that division.

From local fallback to composed agents. Our controlled design isolates fallback decisions on restorable execution units. A local gain may change when an accepted model-produced state becomes the start of later decisions. ScienceWorld uses replayed gold prefixes, and CooperBench census workspaces are unavailable for behavioral rescoring. Their reported endpoints therefore describe local decisions rather than full trajectories (Appendix D.5). Whole-task studies should measure final behavioral success, cumulative acquisition cost, and error propagation through a trajectory. Partially reversible or irreversible actions require a different recovery contract. The boundary of a restorable unit is itself a design choice. Finer checkpoints may limit the state that must be restored but require more frequent checks; coarser units can make a postcondition meaningful while enlarging the state at risk. A composed agent must choose that boundary together with its verification and recovery procedures.

From benchmark checks to general stopping signals. The repair study uses one public benchmark and benchmark-provided examples; the problems may have appeared in pretraining. One public-example asset exposes implementation text, and the observed candidate retries have low output diversity (Appendix A.1). Contamination-resistant tasks, generated checks, and more diverse retries could test whether the selection gain persists. Cost evidence covers one four-configuration Qwen3 portfolio; the Llama probe adds coverage evidence without live cost evidence.

From executor accounting to full-system value. CooperBench and ScienceWorld use a fixed executor-token tariff; repair reports a token ratio. Neither measures complete deployment cost. Recorded attempt-time ratios splice independently run cells under unmatched server loads, so they do not establish live latency savings (Appendix D.4). Whole-task studies should assign policies before execution, measure planning, tools, checking, and restoration alongside executor use, and report end-to-end latency under matched serving conditions.

6 Conclusion

AgentClutch separates the success available in an executor portfolio from the success delivered by its stopping rule and the cost of acquiring it. With repair outputs held fixed, public-example gating recovers 16 test-passing fallbacks blocked by syntax stopping, at higher token cost. In CooperBench and ScienceWorld, a middle executor adds complementary operational outcomes while costing more than the reference work its successful stops avoid. These findings motivate evaluating stopping rules and executor portfolios jointly: complementary outputs do not ensure reliable delivery, and additional coverage does not ensure economical acquisition. The conclusions concern restorable execution units; prospective whole-task evaluation remains necessary to establish deployment-level benefits.

AI USE STATEMENT

Generative AI tools assisted with language editing, reference formatting, implementation and debugging, figure preparation, and analysis of recorded results. Under the authors' direction, they also provided feedback on protocol drafts and assistance with the mathematical formulation of the coverage identity and reach-weighted accounting. The authors determined the research questions and experimental decisions, reviewed and tested AI-assisted code, and checked claims against the available records. Benchmark labels and quantitative results came from the execution and analysis code. A separate AI-assisted qualitative review examined 20 frozen accepted states (Appendix D.5); it did not change benchmark labels and was not a human-agreement study. Numerical validation has the scope and limitations stated in the reproducibility records. No AI-assisted translation or formal proof writing was used. The authors reviewed the AI-assisted work and take responsibility for the final text, claims, and artifacts.

REPRODUCIBILITY STATEMENT

The anonymous supplement includes execution and analysis code, registered protocols, configurations, outcome matrices, and available event records. Its manifest maps original hashes to anonymized review copies; identifying machine paths are removed without changing numerical outcomes. Appendix E.4 links each result family to its inputs. The supplement retains all 536 repair outputs, evaluation/public-test replay receipts, deterministic paired statistics, and the 216 command-normalization records, alongside the census, SW4, Llama, and portfolio receipts and the subsequent verifier-audit validity erratum (Appendix D.5). Offline checks reconstruct the stated recorded-data results, repair intervals, and subsequent descriptive portfolio joins; they do not validate semantic correctness or regenerate every historical analysis. Full snapshots, model weights, and unrecovered fitted-router objects are not included. The Llama point estimates are reproducible from sealed cells; its original interval generator was not recovered. Appendix E.3 gives computational details.

REFERENCES

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Manaal Faruqui, and Mausam. AutoMix: Automatically mixing language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling, 2024. URL https://arxiv.org/abs/2407.21787.

Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. CodeT: Code generation with generated tests. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=ktrw68Cmu9c.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research (TMLR)*, 2024.

Jasper Dekoninck, Maximilian Baader, and Martin Vechev. A unified approach to routing and cascading for LLMs. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025.

Lutfi Eren Erdogan, Nicholas Lee, Sehoon Kim, Suhong Moon, Hiroki Furuta, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. Plan-and-Act: Improving planning of agents for long-horizon tasks. In *International Conference on Machine Learning (ICML)*, 2025.

Eric J. Horvitz. Reasoning about beliefs and actions under computational resource constraints. In *Proceedings of the Third Conference on Uncertainty in Artificial Intelligence (UAI-87)*, pp. 429–444, Seattle, WA, 1987.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. RouterBench: A benchmark for multi-LLM routing system, 2024.

Vansh Kapoor, Aman Gupta, Hao Chen, Anurag Beniwal, Jing Huang, and Aviral Kumar. TRIM: Hybrid inference via targeted stepwise routing in multi-step reasoning tasks. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 3236–3257, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/06b33e277e6c48f361621717c49cde22-Paper-Conference.pdf.

Arpandeep Khatua, Hao Zhu, Peter Tran, Arya Prabhudesai, Frederic Sadrieh, Johann K. Lieberwirth, Xinkai Yu, Yicheng Fu, Michael J. Ryan, Jiaxin Pei, and Diyi Yang. CooperBench: Why coding agents cannot be your teammates yet, 2026.

Ji'an Lei and Jian Huang. TACIT-Switch: Cost-aware model escalation for LLM agents from censored supervision, August 2026.

Songyuan Li, Ahmed M. Abdelmoniem, and Shiqiang Wang. ProgRouter: Online progress-guided orchestration for multi-agent LLM workflows under quality-cost tradeoffs, August 2026.

Niklas Muennighoff, Qian Liu, Armel Zebaze, Qinkai Zheng, Binyuan Hui, Terry Yue Zhuo, Swayam Singh, Xiangru Tang, Leandro Von Werra, and Shayne Longpre. OctoPack: Instruction tuning code large language models. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 7604–7623, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/1ec299a5229034141e58aeded0d0b9de-Paper-Conference.pdf.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs with preference data. In *International Conference on Learning Representations (ICLR)*, 2025.

Rudrendu Kumar Paul and Sourav Nandy. AgentRouter: Heterogeneous model routing for cost-optimal multi-step agentic workflows. In *ICML 2026 Workshop on Planning in the Era of Language Models (LM4Plan)*, 2026. OpenReview 05yAwXIImz.

Stephan Rabanser, Nathalie Rauschmayr, Achin Kulshrestha, Petra Poklukar, Wittawat Jitkrittum, Sean Augenstein, Congchao Wang, and Federico Tombari. Gatekeeper: Improving model cascades through confidence tuning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-0656. URL https://papers.neurips.cc/paper_files/paper/2025/hash/1c23f6ce43a7db43dfe31f26ceabfbfe-Abstract-Conference.html.

Stuart Russell and Eric Wefald. Principles of metareasoning. *Artificial Intelligence*, 49(1–3):361–395, 1991. doi: 10.1016/0004-3702(91)90015-C.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. HuggingGPT: Solving AI tasks with ChatGPT and its friends in Hugging Face. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2408.03314.

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

Simon Yu, Derek Chong, Ananjan Nandi, Dilara Soylu, Jiuding Sun, Christopher D Manning, and Weiyan Shi. Shepherd: Enabling programmable meta-agents via reversible agentic execution traces, 2026. URL https://arxiv.org/abs/2605.10913.

Jieyu Zhang, Ranjay Krishna, Ahmed H. Awadallah, and Chi Wang. EcoAssistant: Using LLM assistant more affordably and accurately, 2023. URL https://arxiv.org/abs/2310.03046.

Ziqian Zhong, Aditi Raghunathan, and Nicholas Carlini. ImpossibleBench: Measuring LLMs' propensity of exploiting test cases. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 124959–124986, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/ca688eb14e29701a11bdba6633186328-Paper-Conference.pdf.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62138–62160. PMLR, 2024. URL https://proceedings.mlr.press/v235/zhou24r.html.

Appendix roadmap. The controlled repair protocol, retained-output accounting, and independent replay appear in Appendix A.1. The remaining sections provide benchmark definitions, allocation controls, verification diagnostics, and reproducibility information. Table 4 locates each topic; Table 27 summarizes the status of studies supporting main-text claims.

Table 4: **Where to find each kind of detail.**

Question	Section
How were repair outputs retained, selected, and independently tested?	A.1
How were populations drawn, and what is a scored unit?	A.2, A.3
How is each attempt started from the same state?	A.4
What exactly does acceptance check?	A.5
How are intervals computed?	A.6
What were the full results per benchmark?	B
Do the comparisons survive matched attempts and ablations?	C.2, C.5
Would another order or executor set have done better?	C.4, C.6
Does acceptance track implementation correctness?	D.1, D.5, A.1
Do recorded outcomes transfer to fresh execution?	D.2, D.3
What was registered, what deviated, and what is released?	E

A Evaluation setup and measurement

A.1 Controlled Python repair: protocol and endpoint audit

Population and executor information. The study adapts the Python repair tasks in HumanEvalPack to a bounded single-file editing interface. Of 164 problems, 158 passed the recorded eligibility controls: canonical code passed evaluation and public-example tests, buggy code failed evaluation tests, and a callable public-example check was available. Six lacked a compatible example-check function and were excluded before model calls. A seeded split assigned 24 problems to development and 134 to evaluation. These are held out from our experiment development, not necessarily from model pretraining. The development screen preceded the evaluation matrix. Aggregate development and evaluation results are kept separately.

Each problem has four retained attempts, C_1, C_2, C_3, R_1, initialized from the original buggy implementation. The actor receives the function declaration, specification, buggy code, and public examples. Its tools edit `solution.py` through a local OpenHands-compatible interface and signal completion; the interface does not expose arbitrary file reads or a shell. The recorded configuration is Qwen3-14B with thinking disabled for C and enabled for R, temperature 0.2, top-$p = 0.95$, at most six turns, and at most 8,192 generated tokens per turn. Attempts use separate seed offsets. A generation call without a tool call ends an attempt. The outputs are separately acquired records; policies in Table 1 are reconstructed selections over them, not fresh live cascade traces.

Stopping and evaluation. The syntax gate requires a nonempty code change, a parseable abstract syntax tree, and the expected function name. The example gate also requires the public-example check to pass. The separate evaluation-test file and canonical-solution field are omitted from actor prompts and tools. One public-example asset, Python/119, nonetheless contains reference implementation text before its test driver; this actor-visible contamination is analyzed separately below. The coordinator loads both data files, then scores each retained output in a fresh Python subprocess with a five-second timeout and a canonical positive control. This is an information boundary in the actor interface, not filesystem or network isolation: the evaluator implements neither a security sandbox nor separate globals for candidate code and tests. No evaluator result is returned as feedback to subsequent attempts. The endpoint is passing these tests, not unrestricted semantic correctness.

Recorded policy accounting. For binary gate decisions g_e and evaluation labels B_e, let $z_R = g_R B_R$ be gate-filtered reference success and $S_{e\to R} = g_e B_e + (1 - g_e) z_R$ be delivered success from a candidate e followed by the reference. Fixed-output two-stage selection gives

$$S_{e\to R} - z_R = g_e(B_e - z_R) = \mathbf{1}\{g_e = 1, B_e = 1, z_R = 0\} - \mathbf{1}\{g_e = 1, B_e = 0, z_R = 1\}.$$

Taking expectations gives the gate-filtered-reference contrast. This baseline filters the reference through its gate. Against the raw reference evaluation label, the corresponding identity is

$$S_{C\to R} - B_R = g_C(B_C - B_R) + (1 - g_C)(g_R - 1)B_R.$$

The second term records a reference evaluation success rejected by its own operational gate. The repair results use the observed raw-reference comparison directly.

We select the first gate-accepted output and count it as successful only if its evaluation tests pass; a policy rejecting every attempt fails. Every reached attempt contributes all recorded prompt and completion tokens. Raw C_1 totals 425,273 tokens. For raw R_1, the denominator is 436,455 tokens. The syntax-gated $C_1 \to R_1$ path uses 439,960 tokens; the example-gated path uses 568,046. Example-gated $C_1 \to C_2$, $C_1 \to C_2 \to C_3$, and $C_1 \to C_2 \to R_1$ use 540,613, 654,380, and 683,386 tokens. The final analysis sorts task IDs and jointly resamples all arms and gates within each problem: 10,000 paired PCG64 draws, seed 20260923, with linear percentile quantiles and ratios of token sums within draws. Earlier transcription and interval corrections are documented in the supplement; no retained output or evaluation label changed.

Table 5: **Complete retained-output policy comparison on controlled Python repair.** C_1, C_2, C_3 are repeated attempts with the thinking-off candidate configuration; R_1 is the thinking-on terminal reference. Success means selected-output evaluation-test pass on 134 problems; intervals are paired problem-bootstrap percentiles. Tokens charge every reached attempt and are normalized to one raw R_1 attempt per problem. The main-text role names correspond to C_1 and R_1. Candidate retries have low observed diversity ($124/134$ identical C_1/C_2 sources).

Selection policy	Gate	Test pass	95% CI (%)	Tokens/R_1
Raw C_1	None	76.1%	$[68.7, 82.8]$	0.974
Raw R_1	None	68.7%	$[60.4, 76.1]$	1.000
$C_1 \to R_1$	Syntax	78.4%	$[70.9, 85.1]$	1.008
$C_1 \to R_1$	Examples	90.3%	$[85.1, 94.8]$	1.301
$C_1 \to C_2$	Examples	76.1%	$[68.7, 82.8]$	1.239
$C_1 \to C_2 \to C_3$	Examples	76.9%	$[69.4, 83.6]$	1.499
$C_1 \to C_2 \to R_1$	Examples	90.3%	$[85.1, 94.8]$	1.566

Against raw R_1, example-gated $C_1 \to R_1$ gains 21.64 percentage points (95% CI $[14.93, 29.10]$) at a token ratio of 1.301 (CI $[1.182, 1.425]$). Changing the gate from syntax to examples gains 11.94 points (CI $[6.72, 17.91]$). At two attempts, switching from $C_1 \to C_2$ to $C_1 \to R_1$ gains 14.18 points (CI $[8.21, 20.15]$) for 1.051 times the tokens; at three attempts, switching from $C_1 \to C_2 \to C_3$ to $C_1 \to C_2 \to R_1$ gains 13.43 points (CI $[7.46, 20.15]$) for 1.044 times the tokens. These comparisons match attempt limits rather than token budgets.

Table 6: **Joint evaluation-test outcomes for the retained C_1 and R_1 outputs.** Each of the 134 problems contributes one output from each configuration; counts describe availability before a stopping gate selects an output.

	R_1 passes	R_1 fails	Total
C_1 passes	73	29	102
C_1 fails	19	13	32
Total	92	42	134

Table 7: **Candidate gate decisions against evaluation tests.** All 134 evaluation problems, using C_1. A false accept is a gate pass with an evaluation failure. A blocked passing reference additionally requires R_1 to pass evaluation; all such R_1 outputs also pass their operational gate.

Gate	Accept/pass	Accept/fail	Reject/pass	Reject/fail	Blocked R_1 pass
Syntax	102	28	0	4	16
Examples	102	2	0	30	0

The example gate rejects 30 failing candidates and reaches R_1 on those problems, recovering 19 test-passing outputs. The two accepted failures remain failures under R_1 as well. Thus the observed zero reference displacement is compatible with imperfect gate precision. Across configurations, C_1 passes 29 problems that raw R_1 fails; under syntax gating, 16 reference successes are lost, leaving a net gain of 13. Under example gating, none is lost and the net gain is 29. Among repeated candidates, C_1 and C_2 retain byte-identical source on $124/134$ problems. This describes output diversity under the tested sampling configuration and limits claims about retries in other regimes.

The independent receipt audit requires all 134×4 cells, checks record identities, positive controls, token sums, and equality between saved source files and receipt code, and hashes every input. It executes neither candidate code nor new model calls. Historical runner summaries omit the terminal gate when summarizing two-stage success; both the manuscript's analysis and the independent audit apply that gate. The selected-output and cost results above use the latter definition.

Generation provenance and retry diversity. An offline reconstruction applies every retained editor operation to each problem's initial source; all 536 final sources match their saved files. The current runner serializes distinct seeds, temperature 0.2, top-$p = 0.95$, and the thinking toggle. Receipts omit raw requests/responses, request IDs, rendered chat templates, and termination reasons, so they cannot establish the server-effective sampling settings or exclude response reuse. C_1 and C_2 have identical source on $124/134$ problems and identical editor traces on $75/134$.

R_1 leaves the original code unchanged on $36/134$ problems, all recorded as evaluation failures. Thirty of these attempts invoke the finish tool; six terminate without a tool call on the first turn. Three of the six record exactly 8,192 completion tokens; a missing finish reason prevents classifying them definitively as truncated. These outcomes combine repair quality with edit omission and interface termination. They support a comparison of the tested configurations, rather than a general ranking of thinking modes.

Public-example asset correction. Separate-namespace execution revealed that Python/119's public-example asset starts with two indented implementation fragments before its test driver. The original concatenating harness placed these after an unconditional return in the candidate function. An AST check verifies this unreachability for the canonical code and all four retained outputs. The independent public-test replay prefixes a never-called function declaration to contain those fragments; every original byte and every test assertion remains present. The original asset, failed strict control, adapter, and transformation hashes are retained. This execution correction does not undo the implementation text's exposure in the actor prompt. The primary matrix retains the prespecified 134 problems; a separately labeled post-hoc sensitivity removes Python/119 to assess that exposure.

Independent retained-output replay. A second evaluator captures the intended candidate before loading tests into a separate namespace. Each check runs in a fresh digest-pinned Python 3.11 container with networking disabled, a read-only root, an unprivileged user, resource limits, and a five-second execution timeout. Only the read-only worker is mounted; each cell's code and tests arrive on standard input. Assertions must be enabled, the intended candidate must be called, and the driver must complete before a passing receipt is accepted. Eight driver probes cover early exit, candidate-side checker replacement, syntax failure, missing candidate invocation, timeout, and correct/incorrect outputs.

All 402 native controls passed before retained-output execution: canonical evaluation and public-example success, and buggy evaluation failure, for all 134 problems. Replaying all 536 evaluation tests and 536 public gate checks produced zero label changes and no infrastructure failures. Public gates retain the recorded modification/AST conjunction and use the explicit Python/119 adapter above. The original policy counts and regenerated intervals therefore apply.

Removing Python/119 in a post-hoc sensitivity leaves 133 problems and 120 example-gated successes (90.2%), matching the remaining observed C_1/R_1 union. Syntax selects 105 (78.9%); the gate difference is 11.28 percentage points (95% CI: 6.02 to 17.29), with 15 gains and no losses. Relative to raw R_1 (91 successes), the difference is 21.80 points (CI: 15.04 to 28.57) at 1.297 times the tokens (CI: 1.173 to 1.425). This sensitivity changes no original eligibility assignment or primary-matrix record.

Changing syntax to example gating spends 128,086 additional tokens for 16 additional passed problems, or 8,005 served tokens per added solution on the primary matrix. Checker runtime is excluded. The three-attempt switching policy gains 19 problems and loses one relative to homogeneous retries; its aggregate advantage is not set dominance.

This section defines the evaluation populations, scored units, restoration procedure, acceptance rules, and uncertainty estimates used in Appendices B–D. Algorithm 1 gives the fixed-order policy.

Table 8: **Evaluation map.** The studies share a restoration contract but answer different questions with different success endpoints.

Study	Evaluated unit	Main endpoint	Principal purpose
Controlled Python repair	134 repair problems with retained outputs	Selected-output evaluation-test success	Isolate gate-induced selection loss
CooperBench census	357 restored repository steps	Structural-postcondition acceptance	Measure complementary acceptance and executor cost
ScienceWorld	Milestone intervals from replayed prefixes	Native milestone attainment	Examine acquisition cost in another environment

Algorithm 1 AgentClutch for one plan step

Require: Step s, postcondition ok, ordered candidates $e_1, \ldots, e_m$ excluding terminal reference R
$K \leftarrow 0$
for $e \in (e_1, \ldots, e_m, R)$ **do**
  Restore the registered start of s
  Execute e; observe outcome o and attempt cost c
  $K \leftarrow K + c$
  if $\mathrm{ok}(o)$ **then**
    return ACCEPTED, selected outcome o, executor e, K
  end if
end for
Restore the registered start of s
return REJECTED, K

A.2 Populations

Table 9 lists the principal evaluation populations, their analysis units and sizes. The plan-qualification run supplies the plans for CooperBench draw 1; subsequent evaluation draws are disjoint samples from the same pinned 652-pair benchmark. The Llama probe reuses the prospective census (§B.2). Three ScienceWorld samples were drawn successively, disjoint from one another by (family, variation). Section A.3 reconciles the CooperBench counts against the original pool.

Two further registered CooperBench samples were reserved for a separate framework-compatibility study and are excluded from every population above; no AgentClutch claim uses them.

A.3 Units, exclusions, and endpoints

Sample reconciliation. The CooperBench populations draw successively from one pinned dataset; each draw is disjoint from all earlier ones.

The full draw-by-draw pool ledger is preserved in the uncompiled appendix archive and supplement. Pair identifiers were not reused across draws; the census draws from the 257 pairs left after development and peer samples.

Draw 1's 49 evaluable tasks are a subset of the frozen 50, not a further subtraction from the 602 remaining after it: one of the 50 fails a precondition and is excluded. The census uses the 257 pairs remaining after both peer samples have been removed. Four pairs yield no evaluable step,

Table 9: **Evaluation populations.** Clusters are tasks, episodes, or repair problems. Backbones identify the executors; plan qualification uses 14B execution of 27B-authored plans and scores step-level postcondition acceptance. The recorded planner configuration uses the Qwen3.8-27B-FP8 checkpoint, served as `qwen38-27b`.

Population	Environment	Unit	Scored	Clusters	Backbones	Results
Plan qualification	CooperBench	step	81	49	14B on 27B plans	§B.3
Delegation draw 1	CooperBench	step	80	49	8B, 14B	§B.3
Delegation extension	CooperBench	step	86	60	8B, 14B	§B.3
Prospective census	CooperBench	step	357	253	8B, 14B	§B.1
SW1	ScienceWorld	milestone	145	60	8B, 14B	§B.4
SW3	ScienceWorld	milestone	147	60	8B, 14B	§B.4
SW4	ScienceWorld	milestone	149	60	8B, 14B	§B.5
HumanEvalPack repair	Python	problem	134	134	14B off/on	§A.1
SW1 normalization	SW1 reached	milestone	54	35	8B, 14B	§B.4

leaving 357 *steps* over 253 tasks. This step count is distinct from the 357 task pairs remaining at the preceding stage of the pool reconciliation.

Definitions. A *task pair* is one base repository task's two independently authored features, one of the pairs CooperBench generates per base task. A *conflicting pair* is a task pair as CooperBench defines it: two features assigned to the same base state whose implementations can interact, so that neither a clean merge nor a joint test pass is guaranteed. A *plan step* is one plane-validated action: a precondition, a tool subset, a write set, and a harness postcondition check, validated before any executor call. Q^{step} is the share of evaluable steps whose postcondition passes. A *postcondition* is a grounded harness check run on a disposable copy of the workspace; it is never a model's own report of success, and a diff alone is never an accepted check type.

An *evaluable step* is a plan step that reaches a scored postcondition check. In the development draws, four disjoint classes are excluded: *vacuous* steps, whose postcondition already holds before any executor attempt; steps failing *precondition_false*, whose precondition does not hold on a correctly restored snapshot; *instrument_defect* steps, excluded for a harness fault; and *no_snapshot* steps, excluded because no snapshot chain reaches them. Draw 1 reports 3 vacuous, 1 precondition_false, 0 instrument_defect, and 5 no_snapshot exclusions over 80 steps and 49 tasks. The prospective census and Llama probe retain all 357 frozen steps without outcome-dependent exclusions.

Population hierarchy and development overlap. CooperBench tasks are combinations of features drawn from shared repository base states. In our pinned benchmark metadata, Draw 1 contains 80 steps across 49 pairs, 80 individual features, 24 base tasks, and 12 repositories. The extension contains 86 steps across 60 pairs, 86 features, 19 base tasks, and 11 repositories. Combined, the development populations comprise 166 steps, 109 pairs, 130 individual features, 25 base tasks, and 12 repositories. The prospective census contains 357 steps across 253 pairs, 165 individual features, 25 base tasks, and 12 repositories. Between development and census, pair identifiers are strictly disjoint ($0/109$ overlap). However, 120 of the 130 development features appear in census pairs ($120/165$ census features shared), 24 of the 25 base tasks are shared (with `cb-datasets-t7309` in development only and `cb-react-hook-form-t85` in census only), and all 12 repositories are shared. The census therefore evaluates previously unscored feature pairs within the same repository and feature-pool collection, rather than unseen-repository generalization. Clustering sensitivity analysis confirms that the headline economic findings are robust to this grouping structure: resampling complete base-task clusters (25 clusters) yields a 95% bootstrap percentile interval for the cost ratio of $[0.481, 0.838]$, whose upper bound remains well below 1.0 ($\Delta Q = +0.112$, 95% CI $[0.075, 0.149]$). Leave-one-repository-out sensitivity yields ladder cost ratios between 0.547 and 0.668 across all 12 repository subsets, with zero sign reversals.

Units are not comparable. The framework-compatibility probe described above scores one task pair as the unit; the plan-level delegation runs in Table 12 score one plan step as the unit. Both draw from the same CooperBench population, but a task-pair rate and a step rate are never compared numerically anywhere in this paper.

Endpoint. Throughout, Q denotes registered-postcondition acceptance under the rules of §A.5. It is distinct from semantic satisfaction of the written step, from benchmark feature-test success, and from end-to-end task completion. Section D.5 audits this endpoint.

A.4 Execution interface and restoration

ReplayPlane supplies the matched starting conditions and recorded outcomes used by AgentClutch. A step record identifies the fixed plan instruction, registered start, permitted tools, postcondition, executor configuration, verified outcome, and consumed tokens. The allocation comparison holds the instruction and start fixed while varying the executor.

What an arm–step record measures. For CooperBench, the runner restores the snapshot before each bounded run, starts a new model-message context with the same step instruction, fresh anchor, permitted tools, and per-run budget, and carries no previous run's messages or failure explanation into the next run. It retries only after a plane rejection or tool error, up to two runs. The postcondition is checked once on the workspace left by the *last* run, and the cell cost sums tokens over every run. A "single-arm" or standalone-R baseline selects one such cell per step; it does not necessarily mean one bounded run. The census has $202/1428$ two-run arm–step cells; draw 1 and the extension have $46/320$ and $41/344$. The primary comparisons use complete cells. ScienceWorld records one bounded arm–milestone run from a replayed prefix; repair retains one bounded output-generating run per arm and problem. Both reconstruct stopping from separately acquired outputs, without carrying another arm's messages or reducing a later arm's own budget.

The development-only first-run sensitivity in Appendix C.4 truncates each cell's cost to its first run and recomputes the reference denominator on the same rule. Where a retry was triggered, the first run has no separately checked postcondition; the archived sensitivity assigns it *failure* rather than inferring success from the final cell verdict. This is conservative for first-run acceptance, but not necessarily for the normalized cost ratio. The attempts-matched draw 1 controls use separately collected runs, their recorded pass outcomes, and the costs of reached attempts (Appendix C.2).

Restoration. In CooperBench, attempts use restored repository snapshots, current edit anchors, and external postcondition checks. In ScienceWorld, the adapter replays the recorded action prefix before each milestone attempt and applies native subgoal checks (§B.4). A verified pass stops the policy; failure triggers restoration before the next candidate. Restoration gives every executor the same registered start and thereby licenses matched-outcome comparisons. It does not by itself make recorded and fresh outcomes interchangeable, which §D.2 tests directly.

What the records establish. A complete matched outcome matrix supports the reference-coverage identity in Equation 2 and the reach-weighted cost calculation in §3.3. Live execution additionally requires reference outcomes after restoration to be exchangeable with the isolated reference measurements. The matched records alone do not test that premise; §D.2 evaluates it with fresh reference reruns and finds the paired difference inconclusive at the registered precision. Neither the records nor that rerun establishes uninterrupted whole-task completion.

A.5 Acceptance rules

Table 10: **CooperBench acceptance rules.** Each rule maps a step record to a binary verdict; the final column specifies the empty-check case.

Rule	Accepts when	Empty check list
V_{full}	every recorded check passes	reject
V_{must}	every *path_exists* check and every *symbol_grep* check with `must_match` passes	accept
V_{any}	at least one recorded check passes	reject
V_{self}	the final attempt reports completion	–
V_{none}	unconditionally	accept

V_{full} defines registered CooperBench acceptance. The other rules are used only in the sensitivity study (§D.1), which reports acceptance under each variant alongside acceptance under V_{full}. ScienceWorld uses its native subgoal checks. V_{must} accepts every cell accepted by V_{full} and also accepts cells with no required checks; V_{full} and V_{any} reject empty check lists. The self-report rule reads the final attempt's completion flag and can disagree with V_{full} in either direction.

A.6 Estimators and uncertainty

Every interval is a 95% bootstrap unless stated otherwise. Clustering names the resampling unit, not the weighting: for CooperBench, *task-clustered* resamples whole feature pairs (the 253 scored pairs), so every step scored under a sampled pair stays together, while *episode-clustered* resamples whole ScienceWorld episodes and *pooled* analyses weight steps uniformly. Base-task (25 clusters) and repository (12 clusters) resamplings are reported as sensitivity analyses rather than as the primary intervals. Paired contrasts are reported in whichever unit was registered for them, so a task-mean paired difference and a pooled step difference are different estimands and are never compared to each other numerically. Resampling used $B = 10{,}000$ draws throughout, with seeds fixed per experiment and documented with the corresponding results and analysis records. Most intervals are BCa estimates. One ScienceWorld quality contrast uses percentile intervals despite a BCa registration; the descriptive census additions use paired task-clustered percentile intervals. Both distinctions are stated with the results. In-scenario and random-baseline brackets occasionally report the 2.5th and 97.5th percentiles of a draw distribution instead of a bootstrap interval; we label them as drawn percentiles.

B Detailed benchmark results

We report the prospective CooperBench census and its mixed-family coverage probe, the two development draws, and the ScienceWorld evaluations. Appendix C provides the corresponding controls and sensitivity analyses.

B.1 CooperBench prospective census

The census draws 257 previously unscored CooperBench feature pairs and scores all four executor configurations on every evaluable step. Four pairs carry no evaluable step, leaving 253 tasks and 357 steps across the same 12 repositories as the development draws. Running four configurations on 357 steps gives $1{,}428$ recorded cells; every expected cell is present and no instrument fault was recorded. The matrix was sealed before any allocation policy was scored against it, so every policy comparison below is a replay over one fixed set of outcomes.

Policies compared. The reference R is the standalone 14B-on arm–step cell. The *fixed ladder* is the registered instantiation of §4.1: 14B-off, then 8B-on, then R. *Cost Mode* is a dynamic program over the same four configurations, price-ordered, using per-step context to choose the candidate set and stopping rule, and terminating every sequence in R. *Coverage Mode* is exhaustive: it tries every configuration until one passes. *Predict-only* selects one configuration per step from the contextual model and never verifies or recovers afterwards. Single-arm rows run one configuration with no fallback. The oracle bound uses known outcomes to pick the cheapest successful configuration and is not realizable.

Because execution stops at the first recorded pass and R is always the last candidate, every R-terminated policy satisfies $Q \geq Q_R$ pointwise on the sealed matrix. The primary endpoint's quality criterion is therefore satisfied by construction and cannot discriminate hypotheses; the genuine empirical test is the pre-registered cost criterion, an upper confidence bound on $K^{\text{tot}}_{\text{policy}}/K^{\text{tot}}_R$ below one, which Cost Mode passes (0.603, CI $[0.500, 0.723]$).

Gates and headline numbers. Single-arm marginals: 8B-off Q=0.681 at price 0.287; 14B-off Q=0.796 at 0.372; 8B-on Q=0.787 at 0.780; 14B-on (reference) Q=0.815 at 1.000. Cost Mode reaches Q=0.913 at $0.603\times$ the reference price. The fixed ladder reaches Q=0.927 at $0.637\times$. Cost Mode therefore has no established joint advantage over the fixed ladder: $\Delta Q = -0.014$ (CI $[-0.034, 0.006]$) with price ratio 0.946 (CI $[0.858, 1.040]$). Coverage Mode and one pre-frozen

static full-coverage order both reach Q=0.941, equal to the observed success union by construction, at $0.658\times$ and $0.647\times$ respectively. That static order is 14B-off, 8B-off, R, then 8B-on: it permits execution after R fails, so it lies outside the R-terminal family whose full-portfolio minimum is 0.652 on the same matrix and denominator (Appendix Figure 4). The cheapest-success oracle bound reaches the same Q at $0.199\times$ but requires knowing outcomes in advance.

Predict-only gives Q=0.793 at $0.397\times$. The gap to Cost Mode, $\Delta Q = 0.120$ (CI $[0.086, 0.157]$), is the empirical size of the verification-and-recovery effect reported in §4.5. It is conditional on the policies' different stopping rules: predict-only can fall below R because it has no verified reference fallback.

Descriptive recovery contrast. A descriptive analysis after sealing evaluates single-candidate AgentClutch, namely 14B-off followed by the reference after a failed check. It succeeds on $324/357$ steps ($Q = 0.908$), against $284/357$ for 14B-off alone, at price ratio 0.581. The pooled paired success difference is 0.112 (95% CI $[0.079, 0.147]$). The fixed ladder's price ratio is 0.637 (CI $[0.519, 0.770]$), with 73 steps reaching its second rung and 42 reaching the reference. These intervals are paired task-clustered percentile bootstrap estimates added after sealing (10,000 resamples, seed 20260915); every resample includes all steps of each sampled task. They are descriptive additions, not registered gates.

B.2 Mixed-family complementary coverage

We tested whether a candidate from another model family adds accepted outcomes to the same sealed reference. A frozen Llama-3.1-8B-Instruct configuration was evaluated on all 357 census steps from 253 tasks under the frozen attempt budgets, registered starts, tool interface, and V_{full} checks. Interface changes were validated on 12 development steps before the census run. All six shards completed, with no instrument-defect exclusion; the outcomes were sealed before the one-time registered analysis. Preflight records flag 14 vacuous postconditions and 9 false preconditions; these are properties of the shared registered step starts and checks, not Llama model outcomes. They were not used to filter either the Llama join or the Qwen census: both retain the registered 357-step denominator. The released aggregate counts do not identify which of the seven census middle-rung rescues, if any, carry a flag; the two flag sets need not be disjoint. The Qwen3-14B-on reference outcomes were reused without rerunning that arm.

Table 11: **Llama candidate joined to the sealed CooperBench reference.** The union and fallback reach are reconstructed from independently recorded cells. Intervals are archived task-clustered BCa 95% estimates ($B = 10{,}000$, seed 20260921); the original interval generator was not recovered and no new interval was estimated for R.

Quantity	Count / steps	Estimate	95% CI
Llama acceptance Q_L	$140/357$	0.392	$[0.341, 0.446]$
Sealed reference Q_R	$291/357$	0.815	–
Accepted union $Q_{L\cup R}$	$309/357$	0.866	$[0.826, 0.898]$
Additional accepts $Q_{L\cup R} - Q_R$	$18/357$	0.050	$[0.030, 0.079]$
Reference reach after Llama failure	$217/357$	0.608	$[0.554, 0.658]$

The Llama candidate accepts 18 steps that the sealed reference rejected (Table 11). Its standalone acceptance is lower than that of Qwen3-8B-off on this census (0.681), but the added coverage shows that complementary accepted outcomes also occur across these two model families. A subsequent descriptive join finds that all 140 Llama accepts already lie in the full four-Qwen union of 336 steps: Llama adds zero acceptances beyond that union and two beyond the frozen ladder's 331. Thus the gain over R does not expand the observed full-portfolio coverage. One candidate on one frozen population does not establish broad cross-family generalization. The union is a recorded-outcome comparison; no live mixed-family fallback chain or fresh reference extension was run.

The Llama run served 3,062,979 input and 345,002 output tokens. Total recorded attempt wall times were 7,614.7 seconds for Llama and 45,185.5 seconds for the separately collected reference cells. These are execution diagnostics, not a live-policy latency comparison. No comparable frozen tariff was available for Llama, so we report no dollar ratio or cross-provider cost claim.

B.3 COOPERBENCH PLAN-LEVEL DELEGATION DRAWS

Two registered populations compare executors on identical 27B plans. Draw 1 has 80 paired steps from 49 tasks; the disjoint extension has 86 paired steps from 60 tasks. The populations are analyzed separately because neither registration specifies a pooled estimate.

Registered design. A rich planner writes a versioned plan into the control plane: each step carries a precondition, a postcondition check, a tool subset, and a write set. Before execution, the plane validates each step's contract against the current workspace. The first claim compares Qwen3-8B against Qwen3-14B (both thinking-enabled) as executors of the identical frozen plans under a byte-identical executor prompt, paired per step from the same plane-restored snapshot. It requires an absolute floor $Q_{14B}^{step} \geq 0.70$ and non-inferiority $Q_{8B}^{step} \geq Q_{14B}^{step} - 0.15$ by task-clustered BCa bootstrap, with $n_{step} \geq 80$ paired evaluable steps and $n_{task} \geq 12$; parse failures count as failures and stay in the denominator. An amendment adds two further arms, 8B-off and 14B-off (the thinking setting is part of each arm's definition, and prompts are otherwise byte-identical across all four arms), together with a second claim: a cost-ordered escalation cascade that keeps only arms with single-arm $Q \geq 0.70$ measured on the confirmatory data itself, escalates only when the harness postcondition check fails, and stops at the first pass. It must be non-inferior to fixed 14B-on on step-level Q at the same margin and have a task-clustered price-ratio upper confidence bound below 1.0, against a registered prediction of 0.70.

The primary economic axis is executor cost under the fixed token tariff. 8B costs \$0.072/\$0.717 per million input/output tokens and 14B \$0.144/\$1.434, so the 8B tariff is half the 14B tariff on both axes. Price depends on both served input and output counts, and a shared prompt template does not imply equal cumulative input volume across attempts. Secondary axes are served tokens, rich-call count, and wall time (including harness postcondition-check time, reported separately and never folded into price). A full-system amortization claim would additionally require the median usable horizon to exceed $H^* = P_{\text{plan}}/(E_R - E_{\text{cascade}})$ by a pre-registered margin, where P_{plan} is planning cost and E_R, E_{cascade} are per-step executor costs; that claim is not made here.

Diagnostic measurements motivating this design. These are unregistered context for the main text. On 62 Qwen3-14B CooperBench trajectories (1,056 agent steps), 329 of 375 tool-error breaks are stale edit anchors, of which 59.3% are same-intent retries that a plane-side fresh-anchor check can resolve; the median usable horizon is $H = 2$ steps, and resolving those breaks raises the upper bound to $H = 4$. This motivates the fresh-anchor support that §C.5 varies. A served-token audit on the same traces finds the fixed prefix is 32.9% of the served prompt under native tool calling against 63.0% under prompt-mode tool use; prefix caching discounts that share but does not reduce token count, and a slim one-step executor call is 0.094 of the full served call.

Plan qualification. The medium-effort 27B planner passed the registered quality floor and planner-dose precondition: its 14B-executed plans passed 64/81 evaluable steps across 49 tasks ($Q_{plan}^{step} = 0.790$), supplying draw 1. The low-effort baseline had only 30 evaluable steps and did not meet the dose precondition. An escalation pilot on those baseline snapshots informed the registered ordering but was non-confirmatory; no claim uses its result.

Table 12: **Registered CooperBench delegation results.** Each draw is analyzed separately. R is 14B-on ($Q = .788$ on draw 1, .837 on the extension); ΔQ is the policy-minus-R task mean. Prices are relative to one R attempt; shown intervals are task-clustered 95% BCa estimates. Reach lists first, second, and reference rung counts for the cascade.

Population	Policy	Q	Paired ΔQ [CI]	Cost ratio [CI]	Reach	Verdict
Draw 1	8B-on	0.825	$+0.022$ $[-0.100, +0.150]$	0.643	–	PASS
80 steps	Cascade	0.975	$+0.165$ $[+0.088, +0.279]$	0.397 [0.254,0.617]	80/14/3	PASS
Extension	8B-on	0.791	-0.046 $[-0.150, +0.050]$	0.682	–	FAIL
86 steps	Cascade	0.919	$+0.092$ $[+0.042, +0.175]$	0.743 [0.489,1.045]	86/22/12	FAIL

Table 12 reports both registered claims on both populations. Both claims pass on draw 1 and both fail on the extension: the tier contrast touches the non-inferiority margin exactly, and the cascade price interval crosses one. The registered contrast is evaluated separately on each draw.

B.4 SCIENCEWORLD EVALUATIONS

ScienceWorld supplies benchmark-authored subgoal instructions and native milestone checks. Each SW1 episode has 2 to 4 ordered semantic milestones (mean 2.42): a milestone marks a change in completed sequential subgoals, co-satisfied subgoals collapse into one milestone, and already-satisfied milestones are excluded. An executor is therefore scored on one interval of a task rather than on the whole task. Three successive populations were drawn, disjoint from one another by (family, variation): SW1 with 145 milestones from 60 episodes across 15 families, SW3 with 147 milestones from 60 episodes over the same families, and SW4 with 149 milestones from 60 episodes. All three use the same command grammar, action cap, token budgets, declared tariff, and registered postcondition checks. SW1 informed subsequent policy development; the SW3 and SW4 policies were fixed before outcomes from their respective populations were observed.

Execution parameters. Each arm replays a stored gold action prefix to the registered start of the interval and then executes at most 90 turns, including truncated turns, with at most one primitive command per turn. Each model call has a 4096-token prompt ceiling and a 4096-token completion limit; an over-budget prompt ends the attempt, while a truncated turn consumes one of the 90 turns. No arm regenerates a gold path. Every arm receives the same frozen 26-template command grammar; observations and milestone instructions are visible, while native scores, goal-progress vectors, gold continuations, and observation-conditioned valid-action lists are hidden. The four arms cross Qwen3-8B and Qwen3-14B with thinking disabled and enabled. The ladder is reconstructed from recorded attempts in the fixed order 14B-off, 8B-on, 14B-on, stopping at the first pass, and the diagnostic union includes all four configurations.

Replay criterion. Replay matches scores and ordered-subgoal states; it does not require byte-identical starting observations. Before aggregation, an exhaustive check re-minted all 145 milestone starts three times. Scores and ordered-subgoal states were stable at every start, while four thermometer observations differed in displayed temperature. Matching is therefore asserted on the registered score and subgoal criterion with cross-arm starting-score reconciliation, not on full environment state. The check changed no scored outcome. Because each milestone starts from a replayed prefix, these isolated starts do not test whether local successes compose through uninterrupted execution.

SW1. Table 13 gives per-configuration outcomes. Quality and price use ratio-of-sums estimators with episode-clustered BCa intervals (10,000 resamples, seed 20260912). The paired quality contrast instead averages within-episode differences, giving $+0.0667$ for the cascade over single-attempt 14B-on with a percentile bootstrap interval $[0.025, 0.117]$; the pooled difference is $10/145 = 0.0690$, a different estimand. The frozen aggregation implementation uses percentile resampling for this contrast, departing from the registered BCa specification, which is why no interval below is interchangeable with one computed the other way. Prices reuse the CooperBench tariff. Planning and verification compute are excluded from declared executor price; planning cost is not measured because the benchmark supplies the milestones.

Table 13: **SW1 executor outcomes.** All arms use the same 145 milestones and 60 episodes. Quality intervals are episode-clustered BCa estimates. Declared price sums input and output token charges over all milestones for a fixed arm, and over consumed rungs for the cascade.

Configuration / policy	Passes	Q^{step} (95% CI)	Price ($)	Cost ratio to R
8B-off	90	0.621 [0.517, 0.706]	0.123	0.100
8B-on	113	0.779 [0.689, 0.848]	1.421	1.152
14B-off	91	0.628 [0.519, 0.721]	0.238	0.193
14B-on	118	0.814 [0.703, 0.889]	1.234	1.000
AgentClutch	128	0.883 [0.809, 0.933]	1.811	1.468

Table 14: **SW1 cost conditional on reaching each rung.** Mean cost is per reached attempt; contribution divides consumed cost by total single-reference cost. First passes count stopping decisions, not unique rescues.

Rung	Reach	Mean cost per reached attempt (USD)	Contribution	First passes
14B-off	145/145	0.001642	0.193	91
8B-on	54/145	0.019686	0.862	31
14B-on (R)	23/145	0.022155	0.413	6
Total	–	–	1.468	128

Matched reached-state diagnostics. A post-hoc comparison holds fixed the 54 milestones where the first rung, 14B-off, fails. The independently recorded 8B-on and R attempts each accept 31 of these milestones. Their total declared prices are \$1.063038 and \$0.842901, respectively: 8B-on costs 1.261 times R on this same subset. Normalizing by population-average reference cost answers a different question. They serve $487{,}867$ and $405{,}047$ input tokens, and $1{,}433{,}628$ and $547{,}123$ output tokens, respectively. The output-token ratio is 2.620; recorded truncated turns number 60 versus 1. Parse-error flags occur on 22 and 24 milestones, so these flags do not establish higher error incidence for 8B-on.

Controlled command-normalization intervention. A fresh 2×2 rerun compared original commands with a deterministic normalizer for the 8B and 14B reasoning-enabled configurations. The 216 attempts cover all 54 reached SW1 milestones, nested within 35 episodes and 11 task families. Each arm replays the same recorded milestone prefix and uses the same model-seed schedule. Starting scores agree in all paired runs; one milestone has differing observation hashes. This comparison therefore matches prefix and score without establishing identical full observations. Table 15 reports the observed outcomes.

Table 15: **A limited command-normalization intervention.** 54 milestones per arm. Intervals are post-hoc 95% episode-clustered percentile bootstrap sensitivities (35 episodes, $10{,}000$ resamples, seed 20260923), preserving the pooled milestone estimand. "Mean actual edits" counts actual command-string edits per attempt, not template matches. Price is mean USD per attempt, computed from recorded input/output-token charges.

Condition	Passed	95% rate CI	Mean turns	Mean actual edits	Mean price
8B original	30/54	$[44.2, 68.2]\%$	29.5	0.00	\$0.0165
8B normalized	31/54	$[45.2, 71.4]\%$	26.2	0.61	\$0.0154
14B original	37/54	$[58.3, 79.5]\%$	20.8	0.00	\$0.0146
14B normalized	35/54	$[53.8, 75.9]\%$	21.3	0.91	\$0.0149

The normalization effect is $+1.85$ percentage points for 8B (episode-clustered CI: -7.55 to 12.07) and -3.70 points for 14B (CI: -12.00 to 3.45). The archived analysis instead resampled milestones independently, with respective intervals $[-7.41, 11.11]$ and $[-11.11, 3.70]$ points; those intervals do not preserve dependence within episodes. Both analyses leave the attainment effect unresolved.

The intervention changed few commands. The archived counter records $1{,}402$ and $1{,}140$ template matches in the normalized arms, including commands left unchanged. Comparing raw and executed strings yields only 33 actual edits on eight 8B starts and 49 on ten 14B starts, all removing parenthetical text. These edits are not established successful parser repairs: the normalizer does not test parseability, and receipts omit per-command environment responses. In the two normalized arms, $1{,}198/1{,}407$ (85.1%) and $925/1{,}149$ (80.5%) actions remain unparseable, respectively. The experiment therefore establishes no benefit for this normalizer; more effective interface repair remains a possible explanation to test.

With original commands, 14B accepts seven more milestones than 8B (12.96 points; episode-clustered CI: 3.70 to 22.41). With normalization, the difference is four milestones (7.41 points; CI: -6.12 to 19.35). These compare complete model configurations; they do not isolate reasoning or state tracking. The fresh original-arm counts also differ from the $31/54$ for each arm in the earlier SW1 matrix. Seven of the eleven within-model outcome-discordant pairs involve no actual command edit. Pairwise outcome changes therefore cannot all be attributed to normalization. Reasoning-token fields do not separately identify reasoning expenditure.

Command-interface failures on SW1. Under the shared frozen interface a material fraction of primitive commands does not parse: pooled rates of unparseable commands range from 0.613 to 0.806 across arms. These outputs are scored as executor behavior and receive no prompt repair, so they qualify the measured cost ordering rather than only its quality. Cap-hit rates are 0.331, 0.103, 0.324, and 0.090 in 8B-off, 8B-on, 14B-off, 14B-on order, and the fractions of attempts containing a truncated turn are 0, 0.207, 0, and 0.014. Failure-set Jaccard overlap is 0.365 between the first two ladder rungs and 0.703 between the two no-thinking arms, so the cheap rungs fail on partly different subsets even though their overall rates are close.

Voided first run. The first ScienceWorld run is void in full because its initial prompt omitted the benchmark's command grammar and produced an interface failure. A later amendment froze a grammar derived mechanically from benchmark action templates and applied it identically to every arm; the reported run reran all 580 cells with fresh model samples, so no trace from the voided run contributes to any result. The episode registry was retained. This repair followed exposure to the earlier run's outputs and is disclosed separately from the pre-scored amendments and serving pins.

SW3. SW3 draws 147 milestones from 60 episodes over the same 15 families on a copy of the SW1 harness, with one model served from a separate endpoint. It was registered before any scored call. The registry was fixed before scoring, and analysis used episode-clustered BCa intervals with $10{,}000$ resamples and seed 20260917 after the outcome records were sealed; the supplementary release identifies the released registries, execution records, and analysis code.

SW3 single-arm acceptance is $94/147$ for 8B-off, $106/147$ for 14B-off, $118/147$ for 8B-on, and $116/147$ for the reference; Table 16 gives the policy-level comparison.

Reconstructed from recorded attempts, the fixed ladder passes $125/147$ ($Q = 0.850$) at \$1.752, a ratio of 1.287 to the reference with 95% interval $[0.664, 1.753]$; rung reach is 100%, 27.9%, and 17.0%. Coverage falls two milestones short of the four-arm union (0.864), both passed only by 8B-off.

A registered secondary test ran a live value-of-computation reordering policy from the registered SW3 milestone starts, requiring an upper confidence bound below one on $K^{\mathrm{tot}}_{\mathrm{VoC}}/K^{\mathrm{tot}}_{\mathrm{fixed}}$ with no loss of acceptance. The policy accepted $126/147$ milestones ($Q = 0.857$) against 125 for the fixed ladder, but its price was 1.025 times the ladder with 95% interval $[0.851, 1.723]$; in absolute terms it cost about 1.319 times the reference, within 1% of the cheapest static order attaining the same high-coverage regime (1.312). The experiment therefore did not establish the registered cost reduction. Because the interval crosses one, this is an inconclusive test rather than evidence of zero effect: under this protocol and portfolio, online reordering neither demonstrably saved nor demonstrably cost more than the frozen ladder.

Table 16 compares policies reconstructed from each population's own recorded attempts. The separate SW3 reordering test above uses live execution.

B.5 Prospective minimal-policy evaluation (SW4)

SW4 is the smallest policy suggested by the recorded analyses run live: one verified 8B-off attempt, followed on failure by restoration and one 14B-on attempt. It was registered before any scored call on a population of 149 milestones from 60 episodes over the same 15 families, disjoint by (family, variation) from the earlier ScienceWorld samples and selected without model calls. Two arms ran live and sequentially: the minimal policy and an independently sampled single-attempt reference on every milestone.

Table 16: **ScienceWorld policy comparison.** Price ratios use each population's single-attempt R =14B-on cost. Intervals are 95% episode-clustered BCa estimates. The SW1 omit-middle policy keeps 14B-off as the first rung and omits only 8B-on; its $122/145$ accepts and 0.8762 cost ratio are reconstructed from matched records. The separate 8B-off $\rightarrow R$ policy also changes the first rung. Both two-rung policies are descriptive; N/R means not reported.

Quantity	SW1	SW3
Standalone R acceptance Q	0.814	0.789
Fixed-ladder acceptance Q	0.883	0.850
Omit-middle (14B-off $\rightarrow R$) acceptance Q	0.841	N/R
Four-executor union Q	0.883	0.864
Fixed-ladder price ratio	1.468	1.287
Fixed-ladder price ratio: 95% CI	$[1.003, 2.032]$	$[0.664, 1.753]$
Omit-middle (14B-off $\rightarrow R$) price ratio	0.876	N/R
Second-executor reach	37.2%	27.9%
Reference reach	15.9%	17.0%
8B-on single-arm price ratio	1.152	1.031
Single-candidate (8B-off $\rightarrow R$) acceptance Q	0.848	0.837
Single-candidate (8B-off $\rightarrow R$) price ratio	0.796	0.798

The registered analysis is episode-clustered BCa on the pooled price ratio ($B = 10{,}000$, seed 20260919). It gives $Q_{\min} = 0.859$ against $Q_R = 0.832$, a point difference of $+0.027$ that meets the frozen -0.05 quality margin. The archived descriptive 95% interval for this difference is $[-0.020, 0.077]$; the registered quality criterion uses the point estimate. The pooled price ratio is 0.744, with 95% interval $[0.435, 1.046]$. The quality criterion therefore holds and the cost criterion does not: the upper bound exceeds one. The interval is too wide to confirm cost superiority, and this result does not identify the source of that uncertainty. Per the frozen stopping rule this result is terminal: the policy was not retuned, no further population was drawn, and the descriptive SW1 and SW3 reconstructions above stand as post-hoc analyses rather than as evidence for the live claim. A pre-launch power check on the sealed earlier populations had estimated $\Pr(\mathrm{UCB}_{95} < 1)$ at 0.939 and 0.844 under their matrix-replay economics and noted that live fallback could add sampling variability.

C Allocation controls and sensitivity analyses

These controls examine attempt budgets, execution support, executor order and inclusion, and pre-execution prediction. Recorded-policy comparisons reuse matched outcomes; retry controls and execution-support interventions use separately collected attempts, as specified below.

C.1 Policy comparisons on matched steps

Tables 17 and 18 re-implement several allocation policies on the same frozen plans, executor pool, starting states, and verifier. Retry controls additionally vary the number of attempts. It does not compare peer implementations, and it is not a registered test. Population 1 uses the draw 1 plans (80 steps, 49 tasks) and population 2 the extension plans (86 steps, 60 tasks). Rows use the benchmark matrices or the additional retry traces described below. The extension has no attempts-matched resamples, so those rows are absent there.

Table 17: **Matched-step policies, draw 1 (80 steps).** Q is pooled registered-verifier success; price is relative to one complete 14B-on arm–step cell per step. Random-policy brackets are draw percentiles; other brackets are 95% intervals. Reach is a fraction of scored steps. Extra accepts are selected outputs accepted by the policy's stopping rule but rejected by V_{full}, divided by all scored steps.

Population 1: 80 steps, 49 tasks.

Policy	Q	Price	Reach stage 2 / 3	Extra accepts
8B-off	0.600 (48/80)	0.347	–	–
14B-off	0.825 (66/80)	0.295	–	–
14B-on (reference)	0.788 (63/80)	1.000	–	–
14B-off, up to 3 tries	0.863 (69/80)	0.450 [0.277, 0.822]	–	–
Matched random	0.823 [0.775, 0.863]	0.444 [0.382, 0.554]	17.5% / 3.75%	n/a
Self-report	0.825 (66/80)	0.298	–	14/80 (17.5%)
Budget cap (threshold 2)	0.850	0.630	25.00% / –	–
AgentClutch	0.975 (78/80)	0.397	17.5% / 3.75%	–

Table 18: **Matched-step policies, extension (86 steps).** Definitions and cost denominator match Table 17; no additional retry traces were collected for this population.

Population 2: 86 steps, 60 tasks.

Policy	Q	Price	Reach stage 2 / 3	Extra accepts
8B-off	0.686 (59/86)	0.297	–	–
14B-off	0.744 (64/86)	0.345	–	–
14B-on (reference)	0.837 (72/86)	1.000	–	–
14B-off, up to 3 tries	not run		–	–
Matched random	0.763 [0.721, 0.802]	0.659 [0.559, 0.785]	25.6% / 14.0%	n/a
Self-report	0.791 (68/86)	0.426	–	18/86 (20.9%)
Budget cap (threshold 2)	0.849	0.804	30.23% / –	–
AgentClutch	0.919 (79/86)	0.743 [0.489, 1.045]	25.6% / 14.0%	–

Table 19: **Execution support and routing signal.** Draw 1, 80 steps and 49 tasks. Q is pooled success and price is relative to single-attempt 14B-on. The two panels are separate interventions; the anchor-off ladder reuses later-rung traces without reapplying the eligibility floor.

Variant	Step success	Relative price
A. Fresh-anchor support		
14B-off, anchor on	0.825	0.295
14B-off, anchor off	0.650	0.313
Fixed ladder, anchor on	0.975	0.397
Fixed ladder, first-rung anchor off	0.988	0.706
B. Routing signal		
Harness-triggered cascade	0.975	0.397
Self-report cascade	0.825	0.298

Two definitions make the numbers compatible. Retry-only and self-report prices use the same single-attempt reference cost as the other rows, giving ratios of 0.450 and 0.298, respectively. The released analyses retain the underlying amounts. The attempts-matched table below instead retains cascade-to-comparator ratios, so the two denominators differ and its price ratios are not comparable with these rows column by column. The random baseline averages $1{,}000$ seeded draws (seed 20260910), and its brackets are the 2.5th and 97.5th percentiles of the draw distribution rather than a bootstrap interval.

C.2 Attempts-matched controls and alternative stop rules

The cascade receives up to three configurations per step while its reference receives one. Three registered controls on the same snapshots isolate the value of diversification from the value of extra attempts: the union of three 14B-on attempts, sequential 14B-on stopping at the first pass, and three attempts of 14B-off with the same stopping rule. Base-arm traces are reused without rescoring. Union-of-three and sequential stopping share a pass rule and therefore a quality interval, but charge different numbers of attempts, so the union cost is an upper bound on the sequential cost.

Table 20: **Attempts-matched controls on draw 1.** Eighty paired steps, 49 tasks. Upper panel: ΔQ is the task-mean comparator-minus-cascade difference and price uses cascade-over-comparator ratios, so its denominator differs from Table 17. Lower panel: separate diagnostic contrast with no interval or non-inferiority verdict.

Comparator	Q (passes)	Cost (USD)	Task-mean ΔQ [95% CI] comparator minus ladder	Cost ratio [95% CI] ladder / comparator
14B-on, union of 3	0.950 (76)	\$1.6422	$+0.002$ $[-0.027, +0.081]$	0.124 [0.084,0.186]
14B-on, stop at pass	0.950 (76)	\$0.7422	$+0.002$ $[-0.027, +0.081]$	0.275 [0.159,0.427]
14B-off, stop at pass	0.863 (69)	\$0.2310	-0.061 $[-0.133, +0.014]$	0.883 [0.636,1.078]

Diagnostic self-report control; no CI or non-inferiority test

Q (passes)	Cost (USD)	Pooled ΔQ	Self-report/harness price
0.825 (66/80)	0.1529	-0.150	0.750

The registered superiority rule asks whether the union exceeds the cascade. Its lower confidence bound is -0.027, so the rule does not fire, and the observed two-sided interval $[-0.027, +0.081]$ contains zero. The conclusion is therefore lower declared cost without a detected quality difference, which is not the same as equivalence and does not satisfy the registered superiority criterion. The 14B-off retry control does meet its non-inferiority criterion, differing from the cascade by -0.061 with interval $[-0.133, +0.014]$, above the -0.15 margin, while its own price interval contains one. Throughout these controls the registered cascade is held fixed at $Q = 0.975$ and \$0.2039, with $10{,}000$ resamples and seed 20260904. These attempts are stochastic resamples at temperature 0.2 without seed control, not independent replications. A synthetic degeneracy check confirmed the resamples are not duplicates of the base traces: predicted union quality would have been 0.7875 with think-triple stage reach $[80, 17, 17]$, against 0.950 and $[80, 17, 9]$ observed.

The self-report cascade escalates only when the model itself reports it has not finished. It reaches the same $Q = 0.825$ as single-attempt 14B-off at 0.75 of the harness-triggered cascade's price. On 14 of the 80 scored steps, self-report accepts an output rejected by the registered verifier. Those 14 disagreements are a different set from the cascade's 14 first-stage escalations. This arm has no confidence interval and no non-inferiority verdict.

Estimator robustness. Across five bootstrap seeds per population (the registered 20260904 or 20260907, plus 20260911 to 20260914), the cascade-price interval endpoints move by at most 0.014: no more than 3.9% of population 1's interval width and 2.4% of population 2's. The upper bound stays below one in population 1 and above one in population 2 for every seed, so the two registered verdicts do not hinge on seed choice.

C.3 Offline robustness of the recorded comparison

Leaving out each of the 12 repositories in draw 1 in turn gives cascade success between 0.969 and 0.987, relative price between 0.353 and 0.433, and strongest-fallback reach between 0.026 and 0.046 on draw 1. Weighting tasks equally rather than steps gives success 0.7942 for fixed 14B-on, 0.8980 for retry-matched 14B-off, 0.9609 for retry-matched 14B-on, and 0.9592 for the cascade, against 0.9750 pooled over steps. The cascade's pooled-step advantage therefore partly reflects that some tasks contribute more steps than others; under equal-task weighting its estimate falls slightly below retry-matched 14B-on. Neither analysis establishes a new non-inferiority claim, and neither pools the two registered draws.

C.4 Ordering, executor sets, and the insertion rule

These analyses replay the sealed CooperBench matrix and the frozen ScienceWorld ledger. No new executor was called and no registered record was changed.

Reached-state insertion cost. For a middle executor M reached on event $\mathcal{F}$ before terminal reference R, let y_M indicate acceptance and c_M, c_R be their attempt costs. Its insertion cost over the population is

$$\Delta K = \mathbb{E}[\mathbf{1}_{\mathcal{F}}(c_M - y_M c_R)] = \Pr(\mathcal{F})\Delta K_{\mathcal{F}}, \qquad \Delta K_{\mathcal{F}} = \mathbb{E}[c_M - y_M c_R \mid \mathcal{F}].$$

Expanding the avoided reference cost gives

$$\Delta K_{\mathcal{F}} = \mathbb{E}[c_M \mid \mathcal{F}] - \Pr(y_M = 1 \mid \mathcal{F})\mathbb{E}[c_R \mid \mathcal{F}] - \mathrm{Cov}(y_M, c_R \mid \mathcal{F}).$$

The covariance term matters when M succeeds mainly on states where reference execution would have been unusually cheap or expensive. An executor can therefore add unique accepted outcomes and still raise total expenditure. For an added operational acceptance rate ΔQ and per-unit net cost ΔK, value v per additional accept offsets insertion when $v\Delta Q > \Delta K$. Expressing v in units of mean standalone reference cost per scored unit, the descriptive thresholds are $v > 2.9$ for the CooperBench census and $v > 14.3$ for SW1. These are operational-acceptance thresholds, not measured behavioral values.

Table 21: **Reached-state outcomes and middle-executor insertion cost.** The cohort contains steps that reach 8B-on after 14B-off fails. Shared, M only, R only, and neither partition acceptance by 8B-on and R. Cost terms are fractions of population-wide standalone-R expenditure; net equals incurred M cost minus avoided R cost.

Population	Reached	Shared	M only	R only	Neither	Incurred	Avoided	Net
CB census	73	24	7	16	26	0.1492	0.0932	+0.0560
SW1	54	25	6	6	17	0.8617	0.2702	+0.5915

Paired insertion uncertainty. As a post-sealing sensitivity, we resample all 253 CooperBench task-pair clusters or all 60 SW1 episode clusters, retaining each cluster's paired prefix, middle, and reference records. Each of 10,000 percentile draws recomputes $\sum_s \mathbf{1}_F(c_M - y_M c_R)/\sum_s c_R$ over the resampled *full* population, including the standalone-reference denominator. The 95% intervals for normalized net insertion cost are $[0.0036, 0.1123]$ in the census and $[0.2578, 1.0335]$ in SW1. Unique incremental acceptance, $\sum_s \mathbf{1}_F y_M(1 - y_R)/n$, is $7/357$ (95% interval $[0.0057, 0.0349]$) and $6/145$ (interval $[0.0073, 0.0800]$), respectively. These quantify sampling variation within the recorded task and episode populations; neither interval assigns a deployment value to added coverage. They resample recorded execution units conditional on retained executor outcomes, so they do not include variation from regenerating model outputs. Resampling the census by its 25 base tasks instead gives a net-cost interval $[0.0077, 0.1040]$. Leaving out one repository at a time gives insertion estimates 0.0355–0.0647; leaving out one SW1 task family gives 0.3854–0.6613. All remain positive on these recorded-data sensitivities. Seeds 20260924 and 20260925 and the released input-to-result script are in the supplement.

Coverage bands. Replaying every subset and order of the three non-reference candidates terminated by R gives 16 sequences per domain and shows acceptance as a function of the executor set alone. Table 21 gives the reached-state outcome partitions and insertion costs behind the main-text comparison.

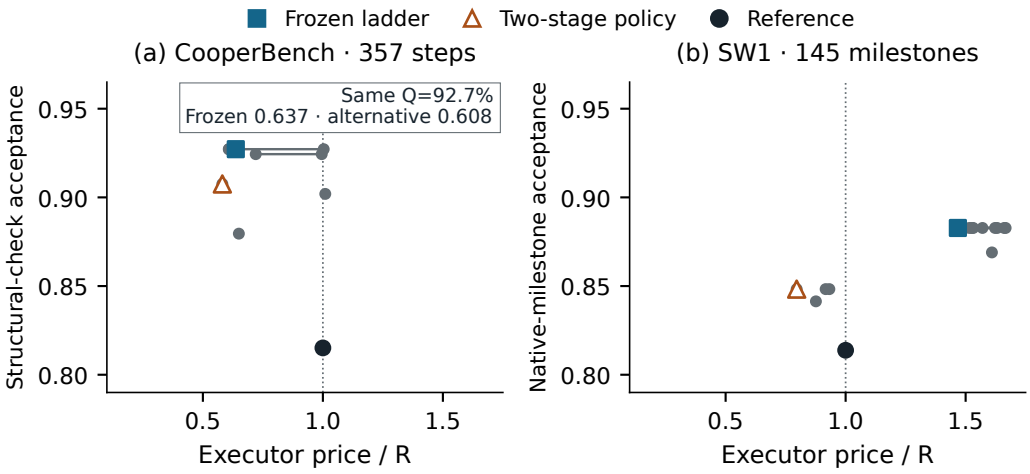


Figure 4: **Executor sets determine recorded acceptance, while order changes cost.** Each panel shows 16 retained reference-terminal sequences. Horizontal segments show variation across executor orders, not confidence intervals, with frozen-ladder, two-stage, and reference policies marked separately. The two-stage candidate is 14B-off for CooperBench and 8B-off for SW1. The labeled CB alternative uses 14B-off, 8B-off, and R: its 0.608 cost ratio and the frozen ladder's 0.637 both accept $331/357$, though their accepted sets differ. CooperBench measures structural-check acceptance; SW1 measures native-milestone acceptance.

Order and selected-output success. Ordering can also change delivered success when the gate accepts a behaviorally failing output. For fixed outputs, order-independent gate decisions g_a, g_b, and the same suffix V, swapping two adjacent executors changes selected-output success by

$$S_{a\to b\to V} - S_{b\to a\to V} = g_a g_b (B_a - B_b). \tag{5}$$

If both outputs pass the operational gate but only one passes independent evaluation, their order changes delivered success although the accepted union is unchanged. This fixed-output identity is a design implication, not an empirical repair order-effect estimate. In a global policy, a preceding prefix adds its reach indicator to the right-hand side before aggregation. Portfolio comparisons therefore need both selected-output behavioral evaluation and the cost of reached attempts.

Equal rates from different accepted sets. On the census, the post-hoc sequence 14B-off, 8B-off, R costs 0.608 times the reference versus 0.637 for the frozen ladder, with both accepting $331/357$ steps. A paired record join gives 326 common accepts and five unique accepts per sequence. The alternative's aggregate price is \$0.062956 lower, a 4.6% saving relative to the frozen ladder's own cost. The comparison is descriptive and has no new interval or superiority gate. It qualifies the development-based eligibility rule without replacing the registered portfolio; semantic equivalence of the two accepted sets is unmeasured.

Permuting and thinning the ladder. Two offline sweeps ask how much the frozen choice matters. Ordering the three rungs in all six permutations leaves acceptance unchanged on both populations (0.975 and 0.919) while changing price: the registered order is cheapest on population 1 at 0.397 (rank 1 of 6) and second cheapest on population 2 at 0.743 against a best alternative of 0.702 (rank 2 of 6). Removing one rung, leaving three two-arm ladders, drops acceptance to 0.900–0.963 from 0.975 on population 1 and to 0.860–0.895 from 0.919 on population 2; only deleting 14B-off raises price above the reference in both populations. Restricting every arm to its first attempt trades a little

coverage for a large price change on one population and reverses that trade on the other: Q moves $0.975 \to 0.963$ and price $0.397 \to 0.470$ on population 1, against $0.919 \to 0.907$ and $0.743 \to 0.661$ on population 2. The comparison uses condition-specific reference costs, and $46/320$ and $41/344$ arm-step cells contain multiple attempts without a separate first-run postcondition verdict. For those cells the first-run sensitivity assigns failure when a retry was triggered, charges only first-run tokens, and recomputes its reference denominator; this does not bound the resulting cost ratio in a fixed direction (Appendix A.4).

C.5 Execution-support interventions

Two interventions remove support that the interface normally provides, to measure how much of the observed economy comes from the memory-management layer rather than from allocation. Turning fresh-anchor support off for single-attempt 14B-off costs quality: the task-mean difference is $\Delta Q = -0.187$ with interval $[-0.304, -0.077]$, and pooled success falls from 66 to 52 of 80 steps, a difference of -0.175; the paired price-difference interval contains zero. Swapping the same disabled variant into the ladder's first rung moves acceptance slightly the other way, Q from 0.975 to 0.988, while roughly doubling escalation and raising price from 0.397 to 0.706: the policy recovers the lost first-rung passes further down the ladder, at the cost those later rungs charge. This intervention replaces only the first rung and reuses later-rung traces; the eligibility floor is not reapplied. Its changed first-rung outcomes produce a different accepted union from the registered ladder.

The second intervention changes how ambiguous edits are resolved, accepting the first textual match instead of rejecting ambiguity. The task-mean difference is $\Delta Q = -0.010$ with interval $[-0.082, +0.034]$, and pooled success stays at $66/80$ in both conditions with three flips in each direction. Ladder acceptance is 0.975 either way and price rises from 0.397 to 0.420. Each intervention required approximately one GPU-hour of execution.

C.6 Predictors, learned routing, and portfolio selection

The remaining adaptive alternatives all replace some part of the fixed ladder's decision with a learned one. None of them establishes a joint advantage over it, and each is reported with the population it was fitted on kept separate from the one it was scored on.

Frozen feature set and predictor. The contextual controller conditions on a six-dimensional feature vector $\phi(s)$: four counts and a maximum pattern length read from the step's postcondition specification, plus the plan-step index (Table 22). All six are available before any executor call and are standardized once against the pooled development population (166 steps), with zero variance mapped to one. Success is modeled per arm by an L2-regularized ($\lambda = 1$) logistic regression fit by 6,000 steps of batch gradient descent over ϕ with arm and failed-arm interaction terms; cost is modeled per arm by a log-linear least-squares fit with an empirical $\text{CVaR}_{0.9}$ log-residual tail correction, $\beta = 1$. Both models were frozen before any prospective data were scored and were never refit, recalibrated, or feature-engineered on it. The routing rule is an exact cost-to-go dynamic program over the three non-reference configurations that compares probing the cheapest-in-expectation remaining arm against stopping at the terminal reference, with no threshold beyond a 10^{-12} tie-break. No isotonic or Platt calibration is applied to the predicted probabilities; the calibration statistics below are diagnostics only.

Table 22: **Frozen contextual-controller feature vector** $\phi(s) \in \mathbb{R}^6$. All components come from the step's postcondition specification and plan position, so they are available before any executor call; standardized on the pooled development population (n=166).

#	Name	Definition
1	`n_checks`	number of checks in the step's postcondition list
2	`n_path_exists`	count of checks with type `path_exists`
3	`n_symbol_grep`	count of checks with type `symbol_grep`
4	`n_other_checks`	row 1 minus rows 2 and 3
5	`max_pattern_len`	maximum character length of any check pattern (0 if none)
6	`plan_step`	step index parsed from the task label's `.stepN` suffix (0 if absent)

Discrimination and calibration. Task-clustered 95% intervals are given with each estimate. For 8B-on, AUROC is 0.555 $[0.470, 0.642]$ with Brier 0.162 and ECE 0.062; for 14B-on, 0.576 $[0.488, 0.662]$, 0.146, 0.051; for 8B-off, 0.575 $[0.506, 0.643]$, 0.214, 0.062; and for 14B-off, 0.571 $[0.493, 0.652]$, 0.157, 0.038. Discrimination is modest and below the on-development figure of about 0.72 for the same model class. This is consistent with the limited gains from the tested controllers, but does not isolate their cause.

Failure census. Across the 329 failed cells of the sealed matrix, the categories are: expected edit absent (missing target) 52.9%; turn-cap timeout (the model never called finish) 24.3%; no successful edit across attempts (no-op) 13.1%; harness errors, meaning unparsed tool calls or transport faults, 6.7%; and stale pattern never removed (stale-symbol) 3.0%. A corroboration test on the three categories with at least 30 cells finds every other arm's conditional success rate below its own pooled rate in every category, with differences from -0.29 to -0.47. The categories therefore flag generally harder steps rather than identifying a specific rescuing arm, consistent with the near-chance discrimination above.

How much of the cost headroom reordering captured. Comparing the fixed ladder with the unrealizable cheapest-success oracle gives $G = K^{\mathrm{tot}}_{\mathrm{fixed}} - K^{\mathrm{tot}}_{\mathrm{oracle}} = \$0.942$ over all 357 recorded census steps, or $\rho = G/K^{\mathrm{tot}}_{\mathrm{fixed}} = 0.688$ of fixed-ladder executor expenditure, above the registered materiality threshold of 0.10. Contextual reordering captures $\eta = 0.079$ of this headroom, below the registered target of 0.50. These descriptive bounds leave substantial potential savings unexplained by the tested reordering rule.

Registered portfolio-construction rule. The registered instantiation screens executor eligibility using development acceptance, then orders eligible configurations by the fixed cost criterion. A separate registered portfolio-selection analysis asks whether an order learned from training folds transfers to held-out repositories or task families: for each held-out fold it fits a price-aware sequence from the training folds' marginal success and cost statistics only, using no per-step features, and scores it against the fixed ladder on the fold's held-out cells.

On CooperBench, leaving out each of the 12 repositories in turn and pooling over $n = 357$ gives $Q_P = 0.9216$ and $\bar{K}_P = 0.0037899$ against $Q_F = 0.9272$ and $\bar{K}_F = 0.0038367$ for the fixed ladder; quality stays within the registered margin and the point estimate favours the rule by about 1.2% on cost. The registered cost criterion is not met, because the lower confidence bound on $\bar{K}_F - \bar{K}_P$ is -3.515×10^{-4}, roughly 7.5 times the magnitude of the point estimate on the negative side. On SW1, leaving out each of the 15 families and pooling over $n = 145$ gives $Q_P = 0.8552$ against $Q_{\text{reference}} = 0.8138$, which holds by construction since the selected sequence is reference-terminated, with a price ratio of 1.1204 and an upper confidence bound of 1.5751 against a required value below one: the registered cost criterion is not met. That ratio is nonetheless an improvement on the fixed ladder's population ratio of 1.468, achieved by demoting 14B-off in favour of 8B-off on exactly the two families that drive the ladder's premium (from $1.70\times$ the reference there against $1.047\times$ on the other thirteen families, roughly halving the local premium to $1.16\times$); selection recovers the pre-frozen static order from marginal statistics alone on 13 of 15 folds.

Neither gate passes; the candidate set, fold definition, and reordering rule were not retuned. The point estimate remains above one on SW1 and its upper bound does not establish a saving; with only 6 to 16 cells per family, the result does not rule out price-aware selection as a principle either. A separate sensitivity run that used the development cells was set aside: a fold-partitioning defect leaked 166 development cells into the held-out splits, which an exact per-repository cell-count reconciliation confirmed, so its numbers are retained on disk but excluded from every claim here.

Historical learned-router comparison. The recorded router aggregates remain in the supplement, but the original fitted objects and per-step selections were not retained. Re-fitting would not recover the reported selectors. We make no primary inference from those aggregates; the six frozen features above define the tested controller.

D VERIFICATION, RESTORATION, AND EXECUTION RELIABILITY

This section examines sensitivity to the acceptance rule, agreement between recorded and fresh outcomes, the effect of restoration, and execution repeatability. It then reports a qualitative source review and explains why two executable verifier audits cannot support their original endpoints.

D.1 VERIFIER-STRENGTH SWEEP

Re-scoring all $1{,}428$ recorded arm-checks under the four alternative rules of §A.5 gives Table 23. Both rates use all rows as the denominator, so they describe each rule rather than any policy. Requiring only the mandatory accept subset changes very few verdicts; requiring any single match or no match at all changes a large fraction, and trusting the executor's own completion flag disagrees in both directions.

Table 23: **Verifier variants: cell-level rates against the registered verifier.** $n = 1{,}428$ arm-rows; both rates use all rows as the denominator.

Variant	Gate accepts rejected by V_{full}	Missed V_{full} accepts
V_{must}	10 (0.70%)	0
V_{any}	277 (19.4%)	0
V_{self}	227 (15.9%)	25 (1.75%)
V_{none}	329 (23.0%)	0

The policy results in Table 24 use all 357 steps as the denominator for both success measures. Q_{stop} scores the selected cell under the rule used to stop; Q_{full} is acceptance of that same cell under the registered checker V_{full}. Only the mandatory-subset variant preserves the registered policies' Q_{full}. The other variants often stop on states that V_{full} rejects, reducing recorded price while increasing declared acceptance. These policy rates differ from the preceding raw cell-disagreement rates, whose denominator is all $1{,}428$ arm-rows; neither measure is chance-corrected.

Table 24: **Acceptance under the stopping rule and registered checker.** Q_{stop} uses the stopping rule; Q_{full} uses the registered checker on the same selected cell. Neither independently measures semantic correctness. A gap means the policy selected a state that the registered checker rejects.

Policy	Metric	V_{full}	V_{must}	V_{any}	V_{self}	V_{none}
Single-candidate	Q_{stop}	0.908	0.916	0.986	0.997	1.000
	Q_{full}	0.908	0.908	0.798	0.826	0.796
	Price	0.581	0.572	0.393	0.497	0.372
Fixed ladder	Q_{stop}	0.927	0.936	0.986	0.997	1.000
	Q_{full}	0.927	0.927	0.798	0.812	0.796
	Price	0.637	0.612	0.410	0.457	0.372

D.2 FRESH REFERENCE AFTER RESTORATION

Equation 2 lets every policy reuse one recorded reference outcome per step. Live deployment instead draws a new reference attempt. One fresh, unmodified rerun of R on every step where a census policy reached it (104 steps, 68 tasks) tests whether those two are exchangeable. Across all 104 steps, recorded and fresh outcomes agree 76.9% of the time, with an exact McNemar $p = 0.31$ on 24 discordant pairs and Cohen's $\kappa = 0.51$: these measurements describe the reached-step subset, not the full census.

On the 42 steps where the fixed ladder actually reaches R (Table 25), the paired difference, fresh minus recorded, is -0.095 with task-clustered 95% interval $[-0.275, 0.073]$ over 40 task clusters ($B = 10{,}000$, seed 20260915). The result is inconclusive at the registered precision: the interval contains zero, but its half-width of 0.174 exceeds the pre-registered 0.15, so this is not an equivalence finding either. Substituting the fresh outcomes on those 42 steps lowers the fixed ladder's implied live acceptance from 0.927 to 0.916 on the full 357-step census ($331 \to 327$ passes). The

Table 25: **Recorded versus fresh reference outcome on the 42 fixed-ladder fallback steps.** The paired fresh-minus-recorded difference is -0.095 (task-clustered 95% interval $[-0.275, 0.073]$).

	Fresh pass	Fresh fail	Total
Recorded pass	7	9	16
Recorded fail	5	21	26
Total	12	30	42

substitution holds earlier rungs fixed, so it is neither a fresh run of the whole ladder nor an estimate of uninterrupted task completion; it separates the exact recorded-coverage identity from the stochastic premise live use requires.

D.3 Restoration ablation

On the same 104-step population, one fresh no-restore 14B-off $\rightarrow R$ chain continues R from the mutated workspace a failed 14B-off attempt left behind, and is paired against restored-start R on the $F = 63$ steps (59 task clusters) where this run's fresh 14B-off failed. The analysis set is therefore determined by which steps the fresh chain failed on rather than fixed in advance.

Table 26: **Restored-start versus no-restore reference, 63 paired steps.** The paired restored-minus-no-restore difference is $+0.016$ (95% interval $[-0.123, 0.161]$).

	No-restore pass	No-restore fail	Total
Restored pass	22	10	32
Restored fail	9	22	31
Total	31	32	63

The difference, restored minus no-restore, is $+0.016$ with 95% interval $[-0.123, 0.161]$ ($B = 10{,}000$, seed 20260916); mean declared R-attempt price differs by $+\$0.00049$ with interval $[-0.00124, 0.00219]$. Among passing steps, at least one attempt carried a tool error for 35.5% $(11/31)$ of no-restore passes and 31.3% $(10/32)$ of restored passes, and no pass in either arm carried a write-set or plane rejection. We therefore detect no pass-rate or cost difference on this subset, with tool-error incidence descriptively similar. This is not an equivalence result, and it has not been tested in ScienceWorld. Restoration remains part of the execution contract because every executor thereby receives the same registered start, which licenses matched-outcome comparisons, not because it is measured to improve performance here.

D.4 Repeatability and recorded overhead

Same-configuration repeatability on SW3. Pairing each arm's live policy attempts against the matrix's independently recorded outcome for the same milestone gives a repeatability diagnostic for the arms the policy chose to try. Both harnesses use temperature 0.2; the different arm sample sizes limit comparisons of repeatability. 8B-off (the primary arm, $n = 145$) shows recorded pass 0.635 against fresh 0.628, a difference of -0.007 with 95% interval $[-0.021, 0.000]$, agreement 0.993, and Cohen's $\kappa = 0.985$. The remaining arms give $\kappa = 0.904$ for 14B-off ($n = 32$), 0.704 for 8B-on ($n = 24$), and 0.750 for 14B-on ($n = 48$). This is a diagnostic rather than a randomized comparison, and it does not test the terminal-reference-after-fallback premise addressed above.

Verification and attempt wall time. Over all $1{,}428$ census cells, verifier wall time is 0.64% of recorded executor-plus-checker time overall, ranging from 0.38% to 2.22% by arm. Summing pooled wall time over the rungs a path reaches and dividing by the pooled single-attempt 14B-on total gives an attempt-wall-time ratio of 0.482 for the fixed ladder and 0.380 for single-candidate AgentClutch. Per-step wall time (median and ninetieth percentile in seconds) is $13.11/184.98$ for the ladder, $13.11/120.29$ for the single-candidate path, and $82.85/236.79$ for the reference alone. Both candidate paths resolve most steps far below the reference at the median, but the ladder's ninetieth percentile exceeds the single-candidate value because it has one more rung to fall through

on hard steps. Each cell was run independently under unmatched server load, so these ratios splice across cells and are not an end-to-end latency trace.

D.5 VERIFIER SCOPE DIAGNOSTICS AND VALIDITY OF WITHDRAWN AUDITS

The registered CooperBench verifier checks specified file and symbol conditions. Passing those checks, satisfying the written step, and passing feature-level tests are three different endpoints. A bounded source review gives qualitative evidence about that gap; two earlier executable audits cannot support their reported endpoints for the reasons below, although their records are preserved.

Blinded source diagnostic. After sealing, we froze a random sample of 20 states from the 118 historically classified scorable, genuinely single-step accepted states in the separate reference-phase archive (sorted task identifiers, seed 20260921). These states reflect reference-phase exploration and are distinct from the paired executor outputs of the prospective census comparison. The census retains outcome and resource records, but its executor workspaces were cleaned up after execution, precluding retrospective behavioral rescoring on those traces; the separate reference-phase archive must not be substituted for them. Eligibility uses the original plan length, not the final committed-step index, and selection did not use the true or false feature-test result. An AI-assisted reviewer read the feature request, frozen step and checks, committed diff, and source context, with executor identity, price, and case-level test outcome withheld; initial annotations were hashed before unblinding. All 40 base and committed file comparisons matched the corresponding original snapshot archive members byte for byte, and no benchmark or model was rerun.

The review flagged local implementation defects in all 20 sampled changes. Examples include an added parameter with no implementing logic, declarations inserted inside an import list, and a new Rust argument never used for the requested fallback. In the last case the required fallback string already occurs in another function, so a file-wide symbol match does not establish behavior in the edited method. Accepted structural conditions can therefore coexist with defects in the written step itself, and a difference in task scope alone does not account for these cases. This is a single AI-assisted qualitative review, not a human-validated semantic success rate: the sample stays fixed despite the infrastructure issues found later, and its annotations do not repair missing test outcomes.

Feature-test overlap: rates withdrawn. The original state ledger contains 217 committed states, reports six infrastructure exclusions, and records nine feature-test passes. Its $9/211$ overlap rate and corrected single-step $7/118$ rate are withdrawn. A later log review found 13 dependency-download and network failures misclassified as test failures, all within the single-step subset: the runner recognized only selected infrastructure conditions, so these failures fell through to the test-failure category and bypassed its retry loop. Other preserved tails lack enough information to establish the cause of the nonzero exit. Removing only the 13 definite errors would give $9/198$ and $7/105$ arithmetically, but would not establish repaired endpoints, so no rate or interval from that reclassification is reported. This supersedes an earlier predicate-only correction.

Mutation sensitivity: not established. The historical ledger records 46 materialized mutations, 36 retained by its classifier, and no recorded escapes across eight parent states. The $36/36$ detection result, the $0/8$ escape result, and the associated binomial and bootstrap bounds are withdrawn. The runner invokes the feature benchmark on the mutation workspace and then invokes the verifier on that same path, without a separate oracle copy or restoration of the mutant between the two; all six inspected benchmark images install exit cleanup that resets and cleans that workspace. Historical image pins, per-mutant oracle output, and before-and-after workspace identifiers were not retained, so the records do not establish that the verifier saw the benchmark-tested mutant. The test-failure classifier also carries the infrastructure fall-through defect described above. Neither behavioral retention nor mutation sensitivity is established by these records; this is an invalid endpoint rather than a result of no sensitivity.

Disposition. Earlier voids for workspace contamination, materialization, and unpacking remain disclosed, as does the nine-seed count observed during debugging. The later validity finding is additive: it changes no original record, executor result, registered protocol, or verifier. No replacement audit was run and no verifier was adjusted on the reviewed examples. The census allocation results and the ScienceWorld cost analyses use separate recorded matrices, so none of these withdrawals

affects them. The archived protocol inventory and released erratum distinguish the historical registrations from the withdrawn endpoints. Because two attempted audits failed on implementation defects rather than on outcomes, semantic correctness of the CooperBench accepted states remains open. The separate controlled repair study in Appendix A.1 supplies test-based behavioral evidence on its own retained outputs.

E Reproducibility and protocol deviations

This section summarizes registered designs, material deviations, cost accounting, and released evidence. The supplement's README locates individual records; its manifest preserves file identities and anonymization provenance.

E.1 Protocol and analysis inventory

Table 27 distinguishes registered experiments, later controlled studies, and subsequent validity findings. Section E.2 collects qualifications that affect interpretation across experiments.

Table 27: **Status of studies supporting main-text claims.** The supplement retains registration and deviation histories.

Study	Status	Endpoint	Material qualification
CooperBench draw 1 / extension	Registered development	Step postcondition; price	Extension price CI crosses one
CooperBench census	Prospective sealed matrix	Step postcondition; price	Pair IDs disjoint; feature and base-task overlap disclosed
ScienceWorld SW1	Recorded transport study	Native milestone; price	Initial interface run voided and rerun
ScienceWorld SW3	Prospective frozen ladder	Native milestone; price	Two-stage reconstruction is descriptive
ScienceWorld SW4	Prospective live	Native milestone; price	Quality criterion met; cost superiority unconfirmed
Controlled repair	Later controlled study	Evaluation-test success	One public-example asset has exclusion sensitivity
Command normalization	Later intervention	Milestone attainment	Attainment effect unresolved

E.2 Deviations that bear on interpretation

The first SW1 interface run was voided after its outputs had been seen; a shared command grammar was then introduced, and all 580 scored cells were rerun with fresh samples. One SW1 quality interval uses percentile resampling rather than the registered BCa method. CooperBench executor choice and order were informed by development populations, which remain separate from the sealed census. Two attempted semantic-verifier audits were withdrawn for implementation defects, so they do not establish correctness of accepted CooperBench states. The SW4 price interval crosses one; its registered cost criterion was not confirmed. Full protocol amendments, void histories, and the stopped SWE-bench development screen are preserved in the supplement and uncompiled archive.

E.3 Cost and compute disclosure

All reported executor prices apply the frozen input and output token tariff to served token counts; they are declared prices, not billed API costs, and no hosted third-party API was used for the reported experiments. Model snapshots and serving configurations are retained in the released records. Planning, restoration, and verification compute are reported separately where measured and are excluded from the executor-price ratios.

The recorded Qwen and Llama runs used locally served models rather than a hosted API. RTX 4090 hardware and vLLM are documented for the CooperBench and repair serving configurations; the

Table 28: **Recorded executor configurations and per-unit limits.** Limits are per model call, not per cell; the CooperBench limits also apply to the Llama candidate joined to its frozen census.

Study	Model configuration	Execution limit and termination
CooperBench	Qwen3-8B or 14B, thinking off/on; Llama-3.1-8B-Instruct probe	Up to two restored runs per arm–step cell; eight model/tool turns per run; up to 8,192 generated tokens per call. Final workspace is checked; all runs are charged.
ScienceWorld	Qwen3-8B or 14B, thinking off/on	One run per arm–milestone; at most 90 turns. Each call has a 4,096-token prompt ceiling and 4,096-token completion limit. An over-budget prompt ends the run; a truncated call consumes a turn.
Python repair	Qwen3-14B, thinking off/on	One run per output; at most six turns and 8,192 generated tokens per call. A response without a tool call ends the run. Historical server-effective sampling settings are unverified.

released ScienceWorld receipts identify model tiers and served tokens but do not supply a complete run-specific hardware inventory. Per-attempt token and available wall-time fields are released, including 7,614.7 seconds of summed Llama candidate attempt time. An experiment-wide GPU-hour total cannot be reconstructed consistently from these records; the declared-price ratios therefore do not represent measured hardware expenditure.

Table 29: **Input-tariff sensitivity of recorded executor prices.** Here a multiplies the fixed input-token tariff, with output tariffs held constant; $a = 1$ is the registered tariff. Ratios use each scenario's own reference cost. Single-candidate is 14B-off on CooperBench, 8B-off on SW1, then R.

Population	Policy	$a = 0$	$a = 0.5$	$a = 1$	$a = 2$
CooperBench census	Frozen ladder	0.409	0.531	0.637	0.812
	Single-candidate	0.357	0.477	0.581	0.753
SW1	Frozen ladder	1.332	1.402	1.468	1.587
	Single-candidate	0.708	0.753	0.796	0.874

Price sensitivity. We froze four descriptive scenarios before recomputation: multiply every input-token tariff by $a \in \{0, 0.5, 1, 2\}$ while holding output tariffs, recorded outcomes, and stopping rules fixed. The 8B to 14B tariff ratio stays at $1/2$; $a = 1$ is the registered accounting. Both policy and reference are repriced from their recorded tokens, so the scenarios show how conclusions would move under a different input-price regime; they do not replace registered endpoints (Table 29).

Overhead sensitivity. Suppose every attempted cell, including the reference-only baseline, incurs the same fee f; with A policy attempts over n steps, a token-price saving disappears at $f = (K_R^{\text{tot}} - K_{\text{policy}}^{\text{tot}})/(A - n)$. At the registered tariffs this crossing is \$0.006784 per attempt for the census ladder (472 versus 357 attempts), \$0.012337 for its single-candidate policy (430), and \$0.004575 for the SW1 single-candidate policy (200 versus 145). The SW1 ladder is already more expensive and makes more attempts (222), so no nonnegative f reverses that ordering. This is a hypothetical fee calculation rather than measured verification or restoration cost, and it is not evidence of full-system savings.

E.4 Code, data, and release contents

The anonymous supplement contains editable source, registered protocols, recorded matrices and receipts, analysis code, subsequent errata, and a manifest of original and released hashes; its README maps each result to files and gives the offline reproduction command. Recorded-data reconstruction checks the headline matrices and retained-output statistics, and independent container replay re-evaluates retained code; fresh model generation requires the documented serving environment and is not reproduced by those checks. Weights and container images are omitted.