A PROTOCOL FOR AUDITING REPLAY-TO-ONLINE TRANSPORT IN RECURSIVE SELF-IMPROVEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Historical replay can identify a promising strategy without showing how that strategy will behave when new tasks arrive. We present a registration and reporting protocol for this replay-to-online boundary. The protocol freezes the candidate set and replay winner before revealing a separately registered live block, limits each tested strategy to the input allowed for the current task, and compares the winner with the incumbent on the same tasks and registered repeats. It reports replay gain ΔR, paired live gain ΔL, and their descriptive difference $B = \Delta R - \Delta L$. A two-world construction shows why the same replay record can be consistent with opposite live gains when replay and live outcomes are otherwise unrelated. The checker rejects duplicate task identities, incomplete replay coverage, and declared protocol violations; unsupported or missing observations remain `UNKNOWN` or `PARTIAL`. We include synthetic checks, a deterministic fixed-policy system check, and a finite model-assisted pilot. These artifacts test the contract and illustrate its reporting boundary; they do not establish predictive validity, population generalization, statistical significance, or online performance.

1 INTRODUCTION

A strategy chosen from historical replay is eventually judged on the tasks it will actually face. Replay can rank a frozen candidate set from recorded histories, but those scores say nothing by themselves about what happens after new inputs, branches, or tool responses appear. Recent recursive self-improvement systems use replay or held-out feedback to guide both selection and later evaluation, so the boundary between the two kinds of evidence matters (Zheng et al., 2026; Nguyen et al., 2026).

Once the winner is chosen, two questions become distinct. Does it outperform the incumbent on a registered block of new tasks? Is it also the best member of the candidate set on that block? The first is the primary comparison here. The second is a diagnostic that requires complete evaluation of every frozen candidate under the same live plan. A winner can improve on the incumbent and still fail to rank first.

The answer also depends on what was fixed before execution and what was actually observed. Repeated sessions can change the weight of a task. Non-matching repeats do not form a pair. Missing outcomes may depend on performance. A hash or freeze flag records content, but it does not show who saw a task or when. The report must therefore distinguish a complete registered block, an observed paired subset, and a comparison that must be withheld.

We make these requirements concrete in a registration and reporting contract. The contract fixes task identity, pairs repeats within a task, and assigns explicit states to incomplete, unsupported, and invalid records. It separates the primary winner-versus-incumbent comparison from rank and regret diagnostics, which require complete scores for every candidate. An offline checker applies the contract to supplied records.

The paper also contains fixed examples for arithmetic and failure handling, a deterministic system check, and a finite model-assisted candidate-selection pilot on two local task families. The pilot is descriptive evidence from a registered finite block. It is not a test of RSI performance, a causal estimate of model benefit, or a population study. Appendix C gives the fixed-policy results separately.

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.

Table 1: Documented mechanisms closest to this work. The table summarizes decision objects and timing; it does not claim that the cited systems or this checker enforce the listed boundaries.

Work	Relevant overlap	Boundary of this paper
Dream-RSI (Zheng et al., 2026)	Replays historical discovery trees to refine exploration policies before redeployment; reports empirical quality and discovery-cost results.	Closest system overlap. This paper neither reproduces nor independently tests those results; it specifies finite-block audit records.
RSEA (Nguyen et al., 2026)	Runs a strict keep-better gate on disjoint evolve/validation data, then evaluates the frozen best on a separate test split.	Shares selection/test separation. This protocol reports paired finite-block winner/incumbent gains and missing-pair states; it adds no update gate or inference guarantee.
SEAL (Guo et al., 2026)	Uses hidden-harness and sealed deployment evaluations; analyzes limits on distinguishing candidates from the available evidence.	Shares evidence-separation logic. The present contribution is a record contract for replay-winner comparisons.
PACE (Shawn, 2026)	Uses paired candidate–incumbent evidence with an anytime-valid test under stated conditions.	Shares paired decisions. The checker implements no e-process, confidence bound or stopping guarantee.
Current protocol	Freezes a replay winner and pairs it with the incumbent on registered post-selection tasks.	Defines finite-block reporting states; complete candidate evaluation also permits descriptive rank and regret.

2 Related Work and Scope

Holdout evaluation separates the data used for selection from the data used for final evaluation, while nested evaluation assesses the full selection procedure (Kohavi, 1995; Cawley & Talbot, 2010). Selective inference conditions error control on the selection event (Lee et al., 2016). Offline policy evaluation targets a policy from behavior data under coverage and modeling assumptions (Jiang & Li, 2016). Missing-data methods connect observed records to a complete target through assumptions about the observation process (Rubin, 1976; Seaman et al., 2013). Our checker borrows the separation discipline but implements none of these estimators or guarantees.

Table 1 places the protocol beside four closely related, documented systems. It compares their decision objects and evaluation timing and stays within what the cited sources describe.

The contribution is a record contract and checker that make candidate freezing, task identity, repeat pairing, support status, and failure handling explicit. The work does not introduce a post-selection, off-policy, or missing-data estimator. Its claims are about the finite-block comparisons supported by supplied records, not about population behavior or deployment thresholds.

3 Audit Target and Measurement

3.1 Selection event and replay gain

Let C be the finite candidate set, including incumbent c_0. Let R denote the registered historical replay units, and let $s_R(c, u, r)$ be the supported score for candidate c, unit u and registered repeat r. Each unit ID maps to one input–truth pair. Reusing a pair under another ID does not create another unit. The replay plan registers every candidate–unit–repeat combination before scoring.

For each candidate, the checker averages valid repeats within a unit and then weights replay units equally:

$$V_R(c) = \frac{1}{|U_R|} \sum_{u \in U_R} \left[\frac{1}{|J_{R,u}|} \sum_{r \in J_{R,u}} s_R(c, u, r) \right]. \tag{1}$$

Selection requires complete, supported coverage of the registered replay plan. The winner is then

$$c^* = \arg\max_{c \in C} V_R(c), \qquad \Delta R = V_R(c^*) - V_R(c_0), \tag{2}$$

with ties resolved by lexicographic candidate ID. This defines the recorded selection event. It does not, by itself, verify candidate bytes or event chronology.

3.2 Paired gain on new tasks

Let L be the separately registered live block. For unit u, let J_u contain the registered repeats with valid, supported scores for both c^* and c_0. Let U_{obs} be the units for which J_u is nonempty. The paired unit difference and observed paired gain are

$$d_u = \frac{1}{|J_u|} \sum_{r \in J_u} \big[s_L(c^*, u, r) - s_L(c_0, u, r) \big], \qquad \Delta L_{\text{obs}} = \frac{1}{|U_{\text{obs}}|} \sum_{u \in U_{\text{obs}}} d_u. \tag{3}$$

We first average within each task and then weight tasks equally. Complete paired coverage describes the registered block; incomplete coverage describes only the observed paired subset. Pairing does not remove performance-dependent missingness or recover the full-block gain from a selected subset.

The protocol also records

$$B_{\text{obs}} = \Delta R - \Delta L_{\text{obs}}. \tag{4}$$

B_{obs} is a secondary descriptive difference between two block-specific gains. It does not separate selection from task composition, support shift, execution noise, or missingness, and it is neither a selection-bias estimate nor a causal effect.

3.3 Complete-candidate selection diagnostics

When the full frozen candidate set is evaluated on the complete live plan registered before reveal, define $V_L(c)$ using the same repeat-then-unit aggregation. The live selection regret and replay winner's rank are

$$\text{Reg}_L = \max_{c \in C} V_L(c) - V_L(c^*), \qquad \text{rank}_L(c^*) = 1 + \#\{c \in C : V_L(c) > V_L(c^*)\}. \tag{5}$$

A tied maximum receives rank one and zero regret. These quantities describe selection quality on the registered block. They require complete all-candidate evaluation; one event cannot estimate a rank-reversal frequency. The winner–incumbent paired gain remains primary.

3.4 What replay alone cannot identify

Assume scores lie in $[0, 1]$ and impose no relationship between replay and live outcomes. Fix any replay table and its selected winner $c^* \neq c_0$. On one live unit, give c_0 score $1/2$. Two worlds consistent with the same replay table give c^* scores $3/4$ and $1/4$, respectively. The live gain is $+1/4$ in one world and $-1/4$ in the other. Replay alone therefore does not identify the sign of live gain under these assumptions.

This construction instantiates the indistinguishable-evidence pattern discussed by Guo et al. (Guo et al., 2026). It clarifies the reporting boundary; it is neither a new impossibility theorem nor an estimate of failure frequency.

4 Registration and Failure Handling

4.1 Information flow and registration

The protocol assigns separate roles to the selector, runner, scorer, custodian, and auditor. The custodian registers the live block and truth before reveal. After the winner is frozen, the runner receives only the input permitted for the current task; the scorer receives the output and sealed truth; the auditor receives the completed records. The checker validates declarations and outcomes. Access control, chronology, and execution still require evidence from outside the checker.

The registration records unit identities, candidate bytes, model or policy version, repeat plan, budget, scorer, initial state, support rule and tie rule. Candidate set and winner remain fixed after replay. Both strategies use the same declared task, scoring version, budget, tool set, randomization and

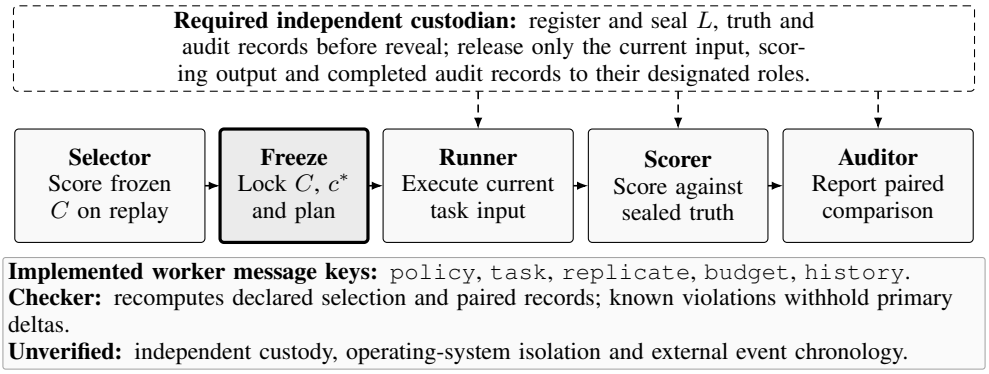


Figure 1: The required sequence is Register → Replay → Freeze winner → Reveal → Pair → Report. The custodian controls the releases between roles. The worker-message contract is implemented; the checker does not verify independent custody, operating-system isolation, or event chronology.

Table 2: Checker states and the comparison each state permits.

Level	State	Meaning for the report
Observation	UNKNOWN	No usable score, including error, timeout, unavailable score or unconfirmed support.
Observation	INVALID	A known protocol violation; retain its reason and withhold primary deltas.
Selection	SELECTED	Complete supported replay permits deterministic selection.
Selection	UNKNOWN	Incomplete or unsupported replay prevents a recomputed selection.
Selection	INVALID	A known replay violation prevents a recomputed selection.
Transport	MEASURED	Complete registered paired block; declared execution conditions remain unverified.
Transport	PARTIAL	A nonempty paired subset is reported as that subset only.
Transport	UNKNOWN	No usable paired unit or support is unconfirmed.
Transport	INVALID	A known violation invalidates the batch comparison.

equivalent initial state. Known state carryover invalidates the primary comparison. A non-resettable environment requires a separately registered design and assumptions.

The local worker accepts exactly the five message keys shown in Figure 1. This message contract is an interface, not an operating-system sandbox. The checker can verify declared hashes and fields, but it cannot show that the live registry or truth remained inaccessible, that freezing preceded reveal, or that submitted candidate bytes match the declarations. A real evaluation therefore needs independent custody, access, event, provider, and reset evidence.

4.2 Failure states and permitted reports

Missing sessions remain distinct from explicit unknown rows. Missing replay coverage withholds selection; a declared winner is retained only as a declaration when the checker cannot recompute it. A declaration that conflicts with complete replay recomputation is rejected. Missing live rows do not undo a valid replay selection, but they prevent a complete live comparison. A nonempty paired subset is reported as PARTIAL; no usable pair is UNKNOWN. Out-of-support observations and unconfirmed support have separate reasons. Known violations such as truth leakage, input overlap, and state interference withhold the batch comparison rather than assigning a score of zero.

An arithmetic report is not a deployment decision. Deployment acceptance would require a registered practical threshold, tolerance, sampling design, and error-control rule; this artifact implements none of them.

Table 3: Counts and results in the fixed synthetic fixture. The pairs are constructed records, not evaluated tasks or independent statistical samples.

Phase	Sessions	Pairs	Groups	Extra	Valid	U/I
Replay	6	2	4	2	6	0/0
Live	4	2	4	0	4	0/0

5 Implementation and Synthetic Verification

The checker recomputes selection, paired unit differences, block-level gains, and status fields from supplied records. It rejects duplicate input–truth identities, incomplete replay coverage, non-finite or oversized numeric inputs, declared protocol violations, and repeated JSON keys. Expected input failures produce standard rejection reports; callers must still check exit codes for output-device failures. Preparation and receipt preflights are separate, and neither one authorizes execution.

The anonymous package contains 59 checker test methods: 27 core, 12 preflight, 11 regression, and 9 input/state-contract methods. Eight fixed counterexample or control inputs accompany expected output fields. One regression method enumerates 81 missing or timeout patterns over four fixture rows; these subcases and test methods are software checks, not research observations.

Here, pairs are distinct input–truth identities, groups are candidate–unit combinations, and U/I counts unknown or invalid observations.

In the replay fixture, $V_R(c_0) = 0.55$ and $V_R(c_1) = 0.80$, so $\Delta R = 0.25$. In the live fixture, the paired unit differences are 0 and 0.2, giving $\Delta L = 0.10$ and $B = 0.15$. The fixed manifest and saved report reproduce these values. This is a test of arithmetic and report construction, not a real task execution.

Appendix C reports a separate deterministic-algorithm system check. Its run-bound values are reported there as exploratory results, not as evidence about RSI models.

6 Finite Model-Assisted Pilot

We also tested the interface with real model calls in a finite model-assisted candidate-selection pilot. Before the first request, we froze the task generator, candidate catalog, prompt contract, and request budget. Each of the 24 events used 4 public-summary inputs, 8 replay inputs, and 8 final inputs; 12 events used max-cut and 12 used knapsack. The 480 task inputs were content-distinct, and the 384 replay/final truth records came from the registered local exact scorer. The event, not a task row or model request, is the statistical unit.

The catalog contained the incumbent and five bounded deterministic policies. Each model selected two IDs from three eligible policies. The model bank and a matched random-pair bank shared the incumbent, two fixed policies, the same replay inputs, and the same final inputs. For event e, we report

$$D_e = V_{L,e}(c_{M,e}) - V_{L,e}(c_{U,e}), \qquad G_e = V_{L,e}(c_{M,e}) - V_{L,e}(c_0), \tag{6}$$

where $c_{M,e}$ and $c_{U,e}$ are the replay winners in the model and random banks. Both contrasts describe this finite block; G_e is not a causal model effect.

Most model-bank contrasts D were negative or close to zero on this registered block. MiniMax had the only positive overall value, but all 24 of its responses failed the strict output contract and fell back to the incumbent. Kimi K3 failed the same contract on 20/24 events. GLM 5.3 had 7 format failures and 2 provider errors; Grok 4.5 and 4.6 each had 5 provider errors. We retained every failure without retry or replacement. Full event receipts, the task registry, recomputed truth records, and the score ledger remain in the workspace; the anonymous package contains the report, aggregate summary, and no-network recheck.

The pilot shows how a real model interface behaves inside a finite candidate-selection audit. It is not a confirmed RSI result. We independently recomputed the local truth registry and scorer without further network calls, but the pilot had no external custodian, independently audited access boundary, or third-party truth endpoint. The result therefore remains descriptive.

Table 4: Finite model-assisted pilot. Each row summarizes 24 event-level observations: 12 max-cut and 12 knapsack. D compares the model candidate bank with the matched random-pair bank on the final block; G compares the model-bank winner with the incumbent. Values are descriptive means over the registered events.

Model	Valid	Format/provider failures	D	G	Regret
Grok 4.5	19	5	-0.002747	0.046155	0.018017
Grok 4.6	19	5	-0.002786	0.046115	0.017719
Grok 4.7	24	0	-0.004947	0.043954	0.020879
GLM 5.2	22	2	-0.007157	0.041744	0.022851
GLM 5.3	15	9	-0.005817	0.043085	0.019154
DeepSeek V4.1 Flash	24	0	-0.009117	0.039784	0.023447
Kimi K3	4	20	-0.000158	0.048744	0.012530
MiniMax M3	0	24	0.000682	0.049584	0.011021

7 Limitations and Required Evidence

The finite pilot used real model providers, but its task registry, truth registry, and exact scorer were local. It had no independently custodied final block, externally audited isolation boundary, or third-party truth endpoint. The local worker's restricted message is a software interface; it does not establish operating-system isolation or event chronology. Those properties require an independently controlled runner and custodian.

All reported quantities are finite-block quantities. A single selection event cannot estimate a rank-reversal rate, and tasks or repeated sessions within an event do not increase the event count. When missingness depends on outcomes, an observed paired subset need not represent the registered block. The checker supplies no confidence interval, significance test, population extrapolation, or deployment guarantee.

The deterministic appendix uses generated algorithmic tasks and a known seed. Its replay and new-task blocks are distinct, and it contains no matched same-task estimate of replay approximation error. Its rank differences therefore do not identify replay error, model capability, or a causal effect. The evaluation count records computation; it is not evidence of cost savings.

The next study should compare replay and direct selection within one frozen candidate set. It should use disjoint input–truth pairs for replay block R, direct-selection block S, and sealed final block F.

Run every candidate directly on the same R tasks under matched conditions to diagnose replay fidelity; these runs do not select a winner. Select c_R by replay on R and c_S by direct execution on S.

Freeze both winners before an independent custodian reveals F. Evaluate c_R, c_S, and incumbent c_0 on matched F tasks with equal budgets, tools, and reset states.

For event e, define $D_e = V_F(c_R) - V_F(c_S)$, using the protocol's repeat-then-task mean.

This procedure contrast differs from ΔL, which compares c_R with c_0. Treat each independent selection event as one study unit; tasks and repeats remain nested within the event.

Record history acquisition, replay, direct selection, final confirmation, failures, and timeouts. Report both the cost from an empty history and the additional cost when an eligible history already exists. List the shared F confirmation cost separately.

Use separate development events to set the event count, maximum quality loss ϵ, minimum cost saving δ, uncertainty analysis, and stopping rule before opening F.

An adoption claim requires the lower uncertainty bound for mean D_e to exceed $-\epsilon$ and the lower event-level cost-saving bound to meet δ. Without a defensible basis for either margin, report descriptive contrasts only.

The replay-versus-direct confirmation study remains planned and unrun. It requires an authorized target, independent custody and scoring of F, verified access controls, and replicated selection

events. The fixed-candidate R/S/F development check and the model-assisted pilot provide engineering and interface evidence; neither satisfies those requirements.

Fixed heuristics can support a fixed-candidate selection study. An RSI claim requires a system that actually generates or modifies candidates.

8 Conclusion

This work specifies when supplied records support a complete or partial comparison between a frozen replay winner and its incumbent on subsequently revealed tasks, and when the comparison must be withheld. Complete all-candidate evaluation also supports a descriptive audit of selection quality. The checker and synthetic records establish conformance on declared inputs. The finite model-assisted pilot shows how provider failures and candidate-bank contrasts appear on a fixed local task block; it did not show a consistent advantage over the matched random comparator. External custody, independently audited isolation, replay-specific error, and practical cost savings require separate evidence.

AI use statement

AI tools assisted with literature search, manuscript editing, and test script preparation. All protocol design, mathematical formulation, experimental decisions, and claims are the authors' work. The pilot executed the registered providers listed in Table 4; the scripts recorded responses and recomputed local task scores.

Ethics statement

The artifact contains locally generated algorithmic task instances and a finite model-assisted pilot; it contains no human-subject evaluation. The pilot used a fixed request budget and local exact truth, but had no independent custody or access review. Known leakage or other protocol violations invalidate the primary comparison; missing observations restrict or withhold it. The artifact does not establish deployment permission, actual isolation, or general model capability.

Reproducibility statement

The anonymous supplement contains the offline checker, preflight gates, 59 checker test methods, fixed synthetic inputs, the unchanged fixture report, eight review counterexamples with expected outcomes, and a reproduction runner. A companion workspace artifact records the 8-model, 24-event pilot, its provider receipts, and an independent no-network recheck. The logical example is specified in Section 3. Reproduction checks the packaged implementation and fixture; it does not validate external authorization, independent custody, or model performance.

References

Gavin C. Cawley and Nicola L. C. Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11:2079–2107, 2010. URL https://www.jmlr.org/papers/v11/cawley10a.html.

Diandian Guo, Cong Cao, Fangfang Yuan, et al. Self-authored verification is unreliable in heuristic self-improving agents, 2026. URL https://arxiv.org/abs/2607.24300v1.

Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48, pp. 652–661, 2016. URL https://proceedings.mlr.press/v48/jiang16.html.

Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 1137–1143, 1995. URL https://www.ijcai.org/Proceedings/95-2/Papers/016.pdf.

Jason D. Lee, Dennis L. Sun, Yuekai Sun, and Jonathan E. Taylor. Exact post-selection inference, with application to the lasso. *The Annals of Statistics*, 44(3):907–927, 2016. doi: 10.1214/15-AOS1371.

Michael Nguyen, Quoc Nguyen, and Paul Vuong. Recursive self-evolving agents via held-out selection, 2026. URL https://arxiv.org/abs/2606.28374v1.

Donald B. Rubin. Inference and missing data. *Biometrika*, 63(3):581–592, 1976. doi: 10.1093/biomet/63.3.581.

Shaun Seaman, John Galati, Dan Jackson, and John Carlin. What is meant by "missing at random"? *Statistical Science*, 28(2):257–268, 2013. doi: 10.1214/13-STS415. URL https://arxiv.org/abs/1306.2812.

Zayx Shawn. PACE: Anytime-valid acceptance tests for self-evolving agents, 2026. URL https://arxiv.org/abs/2606.08106v1.

Tong Zheng, Xidong Wu, Zheng Zhang, et al. Dream-RSI: Recursive self-improvement through evolving worlds, 2026. URL https://arxiv.org/abs/2609.14858v1.

A Implementation Counterexamples

Incomplete candidate coverage. Suppose c_0 scores $(0.6, 0.6)$ on two units, while c_1 has an observed score of 1.0 on only one. An available-case mean selects c_1. If the missing score is 0, the complete mean for c_1 is 0.5 and the selection reverses. The checker withholds selection when registered replay coverage is incomplete.

Duplicate identity. Copying one replay unit under a new ID changes a naive ID-weighted ΔR from 0.25 to approximately 0.266667 because one input–truth pair receives extra weight. The checker rejects the alias instead of treating the second ID as a new unit.

Mismatched repeats. A winner score from repeat 2 and an incumbent score from repeat 1 on the same task do not form a registered pair. The checker averages only common valid repeats and returns `UNKNOWN` when no pair remains.

Shared state. Let the initial environment state be zero. The incumbent returns 0.6 and sets the state to one; a candidate returns the state it receives. Separate reset copies produce a gain of -0.6 in either order. Running the candidate after the incumbent in a shared environment produces an apparent gain of $+0.4$. When the auditor records `state_interference`, the checker withholds all primary deltas. Matching task and repeat IDs alone do not establish equal initial states.

These counterexamples test implementation behavior. They do not establish scientific novelty, model performance, or empirical failure frequency.

B Artifact Details and Reproduction

Both file entry points reject repeated JSON member names at every nesting level. They also reject non-finite numbers and integer literals longer than 4300 digits while retaining the input byte hash and parser reason. Optional authorization and discovery identifiers must be strings or null. Expected parse, validation, and serialization failures produce standard rejection reports; callers must still check the exit code for output-device failures.

Version 2 records preserve registered plans, per-row context hashes, support states, and normalized observations. Reports include source references, execution context, plan hashes, per-unit means, paired differences, absent sessions, and invalid reasons. Preparation checks run with the live block sealed and live rows empty. Receipt checks run only after an authorized execution. A structurally valid report returns `REVIEW_REQUIRED`, with `execution_authorized=false` and `evidence_verified=false`; no checker command authorizes execution.

The anonymous package requires Python 3.10 or later, uses no third-party Python packages, and makes no network requests. From its root, the reproduction runner executes the packaged checks:

```
python -B rsi_transport/reproduce.py
```

The runner checks the 59 packaged test methods, exact fixture-report equality, both template gates, and eight fixed counterexample/control inputs. The fixture command should exit 0. The two template preflights should exit 2 with `BLOCKED`; those refusals are expected for templates without authorized real-run evidence. Test counts describe implementation checks, not study samples.

C Deterministic Algorithm System Check

This appendix reports a system check, not an RSI experiment. It uses generated max-cut and knapsack instances, six fixed deterministic heuristic policies, and an exact local scorer. The seed is known, remote inference is disabled, and the 96 generated inputs cannot serve as a later model holdout. Replay histories were collected on an exhaustive branch-by-step grid rather than by policy-driven exploration.

Each family has 24 replay units and 24 distinct new-task units, with three repeats per unit and one selection event. The replay winner remains frozen. We score all six policies on the new tasks for selection-quality diagnostics. Scores are feasible objective divided by the exact optimum and lie in $[0, 1]$.

Table 5: Decision diagnostics for the run bound to its registered implementation. Each value averages three repeats within each of 24 new-task units and then weights units equally. Each row is one selection event, not a model result.

Family	Replay winner	Live best	Rank	Reg_L	ΔL
Max-cut	`periodic-restart`	`restarts`	4/6	0.007786	0.014137
Knapsack	`periodic-restart`	`restarts`	4/6	0.007603	0.020299

For max-cut, the incumbent, replay-winner, and live-best scores are 0.843265, 0.857402, and 0.865188. For knapsack, the corresponding scores are 0.761770, 0.782069, and 0.789672.

Both replay winners rank fourth among six candidates on their registered new-task blocks and both exceed the incumbent. In both events, `restarts` is the live best. Thus positive incumbent gain and positive selection regret occur together in these two fixed events. The events do not estimate a reversal frequency.

For the same max-cut and knapsack events, replay gains are $\Delta R = 0.023994$ and 0.056216. The secondary descriptive differences are $B = 0.009857$ and 0.035917. Each B compares the registered replay and new-task blocks; it does not identify a selection mechanism, causal effect, or threshold-based acceptance result.

The run records 144 replay trees, 1,728 trajectories, and 50,688 environment evaluations, with zero missing sessions or model calls. It used seed 20260919 and an exhaustive branch-by-step history grid. Because it did not include a same-task replay/direct comparison, its cross-block rankings do not identify replay error. The counts record execution volume, not independent sample size or cost savings. The saved receipt remains bound to its registered implementation; independent chronology, truth custody, and operating-system isolation remain unverified.