Scientific CI/CD for Self-Modifying Discovery Agents: Statistical Gödel Gates, Capacity Budgets, and Domain Verifiers

David Scott Lewis [1]

Abstract

Self-evolving scientific agents can improve by rewriting prompts, tools, workflows, memory, search policies, and eventually evaluators or learned parameters. This capability creates a deployment bottleneck that ordinary benchmark reporting does not address: an edit can improve visible scores while silently degrading calibration, robustness, scientific validity, or future learnability. We propose *Scientific CI/CD*, a governance architecture that treats every self-modification as a high-risk pull request. A candidate edit is promoted only after passing three gates: (i) a Statistical Gödel Gate using anytime-valid e-values or confidence sequences on protected holdouts; (ii) a Capacity Budget Gate that prices model, tool, memory, data-access, and autonomy expansion; and (iii) a Domain Verifier that checks conformal calibration, scientific invariants, and claim-evidence provenance. We define accepted-edit regret, lifetime harmful-adoption rate, evaluator-corruption gap, calibration preservation, rollback utility, and cost-adjusted verified discovery. Three experiments instantiate the gates: a sequential self-edit stream, a capacity-expanding model-selection benchmark, and a dry-lab RNA-seq twin. Across 200-proposal streams, naive repeated testing accepted harmful edits in 6.1% of promoted changes, while the spending-based e-Gödel gate reduced this to 0.008%. Capacity budgeting reduced hidden-risk overfitting in self-expanding polynomial agents, and the domain verifier rejected harmful biomedical workflow edits that passed mechanical or statistical checks. The result is a concrete merge contract for self-evolving scientific agents: improve, but only under valid evidence, bounded capacity, domain constraints, reproducible provenance, and rollback.

[1]AIXC Research, Zaragoza, Spain. Correspondence to: David Scott Lewis <reports@aixcresearch.com>.

The 3rd *AI for Math Workshop* at the 43rd International Conference on Machine Learning (ICML), Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

1. Introduction

Automated research agents are moving from one-shot tool use toward persistent systems that propose hypotheses, execute code, call scientific tools, update memories, and reuse successful procedures across tasks (Lu et al., 2024; Bran et al., 2023; Wang et al., 2023; Romera-Paredes et al., 2024; Yao et al., 2023). The next step is not merely automation but *self-evolution*: the system changes parts of the policy that produce future scientific work. Contemporary examples include self-improving coding agents, open-ended agent archives, skill libraries, prompt compilers, program-search systems, and agentic laboratory workflows (Schmidhuber, 2006; Zhang et al., 2025b; Robeyns et al., 2025; Pourcel et al., 2025; Zhang et al., 2025a; Xu & Yan, 2026; Sureshkumar, 2026). In scientific settings, the same loop can change analysis workflows, domain tools, candidate-evaluation scripts, evidence-ranking policies, and memory schemas.

This is useful only if the changed system remains scientifically trustworthy. Static benchmarks and ordinary regression tests are insufficient because self-modifying systems interact with their evaluators adaptively. Repeated score feedback leaks information, optional stopping inflates false discoveries, and unpriced capacity expansion lets an agent buy apparent improvement by increasing the search class, tool privileges, writable state, or evaluator access (Dwork et al., 2015; Blum & Hardt, 2015; Wu et al., 2025; Ramdas et al., 2022; Wang & Huang, 2026). The dangerous failure is *silent regression*: visible metrics rise while hidden utility, calibration, robustness, or domain validity fall.

We develop Scientific CI/CD, an operational architecture for governing self-modifying discovery agents. The central design rule is simple: a self-edit is not an internal reflection; it is a pull request against the scientific process. The edit may touch prompts, code, tool graphs, memories, objective weights, ranking criteria, learned parameters, or verifiers. It is accepted only if it clears a protected statistical gate, pays its capacity charge, satisfies scientific domain checks, and leaves a replayable provenance trail with rollback pointers.

Contributions. First, we formalize the silent-regression problem for self-modifying discovery agents and distinguish

true utility from proxy improvement under adaptive evaluation. Second, we specify a three-gate merge contract: a Statistical Gödel Gate, a Capacity Budget Gate, and a Domain Verifier. Third, we define six lifecycle metrics for self-evolving science: accepted-edit regret, harmful-adoption rate, evaluator-corruption gap, calibration preservation, rollback utility, and cost-adjusted verified discovery. Fourth, we instantiate the architecture in three experiments covering sequential statistical adoption, capacity drift, and dry-lab biomedical workflow verification. Fifth, we provide algorithms, figures, data, and code in a reproducible package.

2. Problem: Silent Regression Under Self-Modification

Let A_t denote the incumbent scientific agent at step t, and let e_t be a proposed self-edit. The candidate is $A'_t = e_t(A_t)$. The edit can change the research policy π, tool set $\mathcal{T}$, memory M, evaluator J, objective O, or parameter state θ. Let $U(A)$ be protected scientific utility, defined on hidden or prospective evidence, and let $V(A)$ be a visible proxy available during development. A silent regression occurs when

$$V(A'_t) - V(A_t) > 0 \quad \text{but} \quad U(A'_t) - U(A_t) < 0. \tag{1}$$

This gap appears in three ways. *Evaluator overfitting* arises when repeated probing of a finite holdout leaks information (Dwork et al., 2015; Blum & Hardt, 2015). *Capacity drift* arises when an edit expands the hypothesis class or privilege surface without paying a complexity price (Bousquet & Elisseeff, 2002; Dziugaite & Roy, 2017; Zhou et al., 2018; Lotfi et al., 2022). *Domain drift* arises when a workflow remains executable and statistically plausible while violating scientific controls, physical invariants, or claim-evidence provenance (Karniadakis et al., 2021; Missier et al., 2013; Davidson & Freire, 2008; Love et al., 2014).

The following bound illustrates why static checks become brittle. Suppose T candidate edits with non-positive true effects are repeatedly tested on n noisy verifier samples with a one-sided threshold $a > 0$. Even when each estimate is sub-Gaussian with variance proxy σ^2/n,

$$\mathbb{P}\{\exists t \le T : \widehat{\Delta}_t \ge a \mid \Delta_t \le 0\} \le T \exp\{-na^2/(2\sigma^2)\}, \tag{2}$$

and the right side grows with proposal volume. If the agent chooses future proposals based on previous feedback, the fixed-horizon interpretation of each individual test no longer describes the lifetime adoption process. This is precisely the setting addressed by anytime-valid inference, e-values, confidence sequences, and reusable-holdout mechanisms (Ramdas et al., 2022; Wang & Ramdas, 2023; Wu et al., 2025; Sokolova & Sokolov, 2026; Polson et al., 2026; Zampieri, 2025).

3. A Three-Gate Merge Contract

Figure 1 summarizes the merge path. The trust boundary (Gate 0) fixes what the candidate can read, write, and execute. The remaining gates are sequential and fail closed.

3.1. Gate 1: Statistical Gödel Gate

The Statistical Gödel Gate replaces unattainable proof obligations for stochastic scientific systems with anytime-valid evidence. For paired utility differences X_i on a protected verifier, construct an e-process $(E_n)_{n\ge1}$ satisfying $\sup_\tau \mathbb{E}_0[E_\tau] \le 1$ under the null $H_0 : \Delta \le 0$. A simple Gaussian betting process is

$$E_n = \prod_{i=1}^{n} \exp\{\lambda X_i - \lambda^2\sigma^2/2\}. \tag{3}$$

Production deployments should use a robustness stack rather than a single idealized bet: mixture or clipped betting over effect sizes, empirical-Bernstein or self-normalized confidence sequences when variance is unknown, stratified streams for heterogeneous domains, and calendar-time spending audits that detect dependence between proposal batches. The edit is accepted only if $E_n \ge 1/\alpha_t$ for a pre-registered spending rule, or equivalently if a time-uniform lower confidence bound on Δ is positive. The spending sequence (α_t) manages lifetime risk across many candidate edits; hidden-holdout policies restrict query precision, release only significance-gated signals, rotate exhausted tests, and log all verifier access (Dwork et al., 2015; Blum & Hardt, 2015; Globus-Harris et al., 2024; Takahashi, 2026).

3.2. Gate 2: Capacity Budget Gate

An apparent improvement is not comparable if the candidate silently increases capacity. We associate each edit with a capacity vector

$$\mathcal{C}(e) = (d_\theta, d_{\text{search}}, d_{\text{tools}}, d_{\text{write}}, d_{\text{memory}}, d_{\text{data}}, d_{\text{autonomy}}), \tag{4}$$

covering parameter degrees of freedom, search depth, tool privileges, writable surface, persistent memory, data access, and autonomy. For complex privileges, each coordinate is priced by marginal verification cost and blast radius: read-only retrieval is cheaper than executable tools, reversible local memory is cheaper than cross-task memory writes, and evaluator/objective edits pay a multiplier because they alter future evidence. Gate 2 accepts only if

$$\widehat{\Delta}_{\mathcal{H}}(e) - \lambda^\top \Delta\mathcal{C}(e) > \epsilon \quad \text{and} \quad \mathcal{C}(A'_t) \preceq B_t, \tag{5}$$

where B_t is a budget envelope. The charge can instantiate VC/SRM, stability, MDL, PAC-Bayes compression, or domain-specific privilege pricing (Bousquet & Elisseeff, 2002; Dziugaite & Roy, 2017; Zhou et al., 2018; Lotfi et al., 2022; Wu et al., 2026; Nowaczyk, 2025; Zhou, 2026).

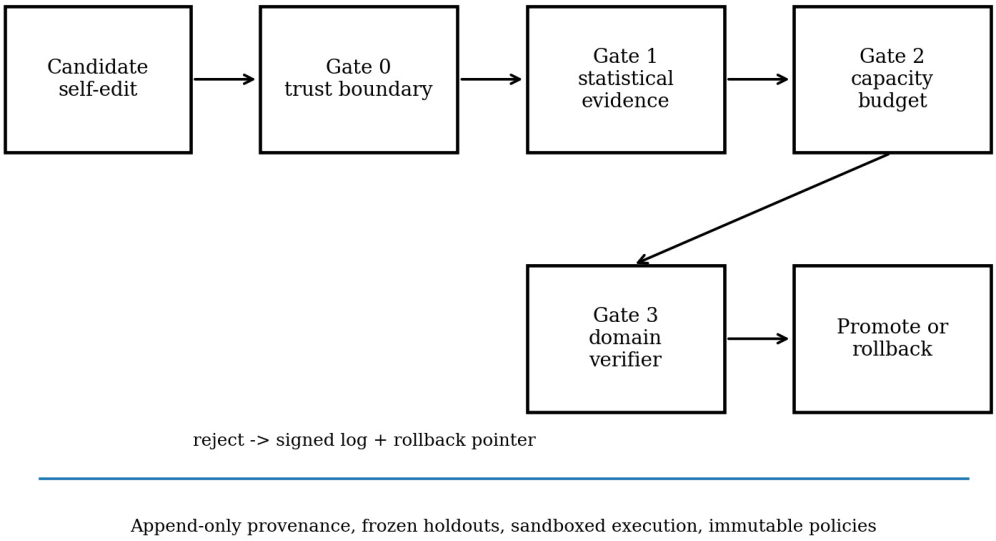

Figure 1. **Scientific CI/CD merge path.** A candidate self-edit passes through a trust boundary, an anytime-valid statistical gate, a capacity gate, and a domain verifier before promotion. Every decision emits immutable provenance and a rollback pointer.

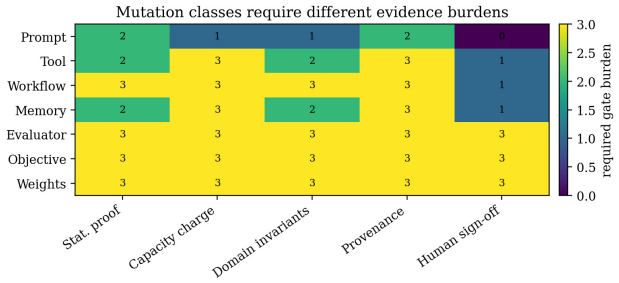

Figure 2. **Evidence burden by mutation class.** The required burden rises as edits become harder to observe, less reversible, more downstream-coupled, or closer to evaluator/objective control. Values are ordinal: 0 absent, 3 mandatory.

3.3. Gate 3: Domain Verifier

Gate 3 asks whether the change remains scientifically valid. A domain verifier combines distribution-free calibration, invariant checks, and claim-evidence provenance. Examples include conformal coverage in predictive workflows (Shafer & Vovk, 2008; Noorani et al., 2025; Detommaso et al., 2023; Kim et al., 2024; Zhang et al., 2026), conservation or residual checks for physical models (Karniadakis et al., 2021; Fasel, 2025; Zhou et al., 2024), biological controls in omics pipelines (Love et al., 2016; Subramanian et al., 2005; Fabregat et al., 2017), and PROV-style artifact lineage (Missier et al., 2013; Davidson & Freire, 2008; Green et al., 2007; Souza et al., 2025).

4. Lifecycle Metrics

The gates are evaluated on the lifetime stream of accepted edits, not only on final benchmark score. For accepted edits $\mathcal{A}$, accepted-edit regret is

$$\mathrm{AER} = \sum_{e \in \mathcal{A}} \max\{0, U(A) - U(e(A))\}. \tag{6}$$

The harmful-adoption rate is the fraction of accepted edits with negative protected utility or later rollback. The evaluator-corruption gap compares proxy gains to protected gains, $\mathbb{E}[\Delta V - \Delta U \mid e \in \mathcal{A}]$. Calibration preservation tracks changes in expected calibration error or conformal coverage. Rollback utility measures recovery time and utility restored after a bad merge. Cost-adjusted verified discovery counts accepted discoveries per total compute, instrument, reagent, failure, and expert-review cost (Tom et al., 2024; Wilbraham et al., 2020; Cao et al., 2026; Xiao et al., 2026).

5. Experiments

We ran three reproducible experiments. While a synthetic verifier does not exhaust scientific risk, it provides a controlled setting for testing whether the proposed gates catch distinct failure modes. The appendix gives full protocols, parameters, additional plots, adaptive-control ablations, cost/rollback audits, and a formal-math domain-verifier instantiation.

Algorithm 1 Scientific CI/CD merge contract

Require: incumbent A_t, candidate edit e_t, protected verifier $\mathcal{H}_t$, budget B_t, domain spec S

Build $A'_t \leftarrow e_t(A_t)$ in a sandbox with write-limited credentials.
Record diff, dependency manifest, tool privileges, random seeds, and rollback pointer in $\mathcal{P}_t$.
if trust-boundary checks fail **then**
  return reject$(\mathcal{P}_t)$
end if
Run paired incumbent-candidate verifier samples and update e-process E_n.
if $E_n < 1/\alpha_t$ and no positive confidence sequence lower bound **then**
  return reject$(\mathcal{P}_t)$
end if
if $\widehat{\Delta}_{\mathcal{H}} - \lambda^\top \Delta\mathcal{C}(e_t) \leq \epsilon$ or $\mathcal{C}(A'_t) \not\leq B_t$ **then**
  return reject$(\mathcal{P}_t)$
end if
if domain verifier $S(A'_t, \mathcal{P}_t)$ fails **then**
  return reject$(\mathcal{P}_t)$
end if
Promote $A_{t+1} \leftarrow A'_t$ and seal the audit record.

Ensure: accepted edit or logged rejection with rollback.

5.1. Experiment 1: Sequential Adoption

We simulated 200 candidate self-edits per run, with true effects drawn from beneficial, null, and harmful mixtures. Each proposal generated up to 80 paired verifier samples. We compared naive repeated one-sided tests, fixed-horizon tests, per-edit e-Gödel tests, and a spending-based e-Gödel rule. Table 1 and Figure 3 show that optional stopping is costly: naive repeated testing accepted 100.5 edits per stream but made 6.17 harmful adoptions on average. The spending-based gate accepted fewer edits (49.2) but almost eliminated harmful adoption (0.005 harmful edits per stream; harmful-adoption rate 0.008%), while increasing the mean true gain of promoted edits from 0.0345 to 0.0522.

5.2. Experiment 2: Capacity Drift

A self-modifying model-selection agent selected polynomial degree for a noisy nonlinear regression problem using a small verifier. Visible-only selection overfit the verifier, choosing degree 15.6 on average with hidden MSE 0.0733 and a generalization gap of 0.0130. A capacity-budget gate selected lower degree (8.7), reduced hidden MSE to 0.0687, and cut the gap to 0.0032. The hidden oracle remained better (0.0614), showing that capacity budgeting is not a substitute for data; it is a guard against unpriced expansion.

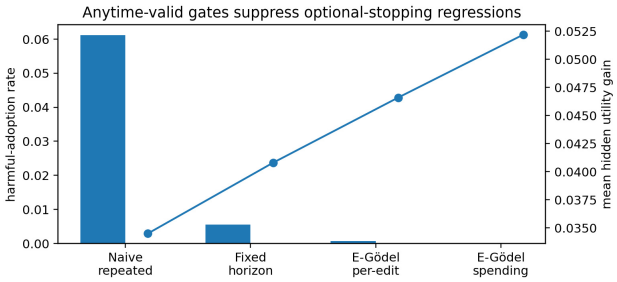

Figure 3. **Experiment 1.** Naive optional stopping promotes many more harmful edits. Anytime-valid e-Gödel rules trade throughput for substantially lower harmful-adoption rate and higher mean protected utility.

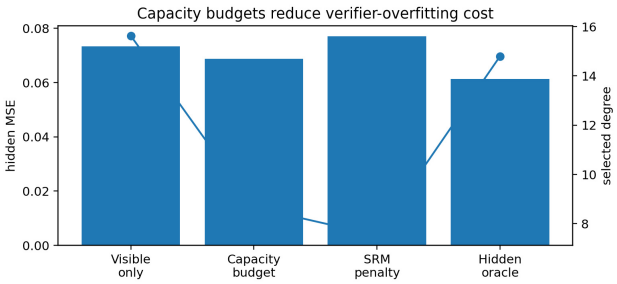

Figure 4. **Experiment 2.** Penalizing capacity expansion reduces hidden-risk overfitting relative to visible-only selection.

5.3. Experiment 3: Dry-Lab Domain Verification

We built a synthetic RNA-seq twin with known positive-control genes, housekeeping genes, negative-binomial count noise, and library-size confounding. Eight workflow variants included benign transformations and harmful edits such as raw-count testing, label shuffling, direction flipping, aggressive filtering, and missing provenance. Mechanical execution passed every variant. Statistical calibration rejected two harmful edits. The domain verifier rejected all five harmful edits, including three that passed the first two gates: shuffled labels, flipped effect directions, and missing provenance. This supports the main claim that domain-specific controls are not redundant with software or statistical checks.

6. Deployment Blueprint

The merge contract becomes useful when it is connected to explicit deployment lanes. A self-editing agent should not receive a single undifferentiated permission to "improve itself." It should receive typed authority over specific mutable surfaces. Table 2 shows the lane structure used in our implementation. Low-coupling prompt or skill edits may be handled by automated gates. Workflow and tool edits require stronger replay and domain checks because they can change the experimental path. Evaluator, objective, data-access, and parameter edits are escalation lanes because they

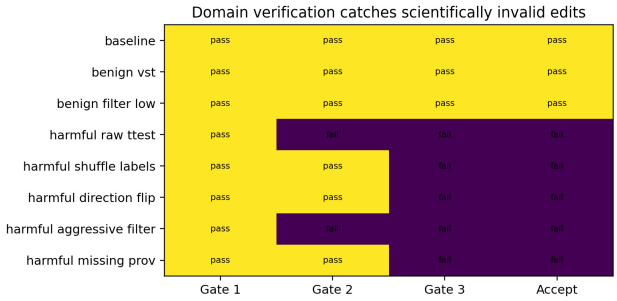

Figure 5. **Experiment 3.** A biological domain verifier rejects harmful workflow edits that remain executable and, in several cases, pass the statistical gate.

Table 1. Compact result summary.

Case	Baseline	Governed
Harmful-adoption rate	6.12%	0.008%
Accepted edits/stream	100.5	49.2
Capacity hidden MSE	0.0733	0.0687
Selected capacity	degree 15.6	degree 8.7
RNA-seq harmful accepted	5/5 possible	0/5

change what future improvements mean or what evidence the agent can exploit.

The key operational principle is *separation of generation authority from acceptance authority*. The agent may generate candidate edits, but it cannot unilaterally modify the holdout, the acceptance thresholds, the capacity ledger, or the domain specification. This separation mirrors secure workflow systems: the process that proposes code is distinct from the process that signs a release. For scientific agents, the same boundary must also cover datasets, assay protocols, theorem statements, pathway libraries, and evaluation scripts. Otherwise the system can satisfy the local text of a metric while changing the ruler that interprets the metric.

Gate 0 as the enforcement layer. Before statistical testing begins, Gate 0 fixes the execution envelope. It hashes the incumbent, candidate, dependencies, verifier version, and data manifests; it grants only the tools required for the proposed edit; and it denies network or filesystem access that is unnecessary for the test. This does not make the candidate safe by itself. It makes failures observable and replayable. A rejected edit should produce a localized diagnostic rather than an anecdote: which invariant failed, which capacity account was exceeded, which holdout stream was touched, and which rollback snapshot restores the incumbent.

Verifier renewal. The holdout lifecycle should be explicit rather than implicit. Static hidden tests eventually become stale or partially revealed. A renewable verifier has six states: committed, sealed, scored, coarsely reported, retired, and refreshed. During scoring, the agent sees only the pass/fail decision or a rounded significance-gated diagnostic. During retirement, the verifier is moved to a post-hoc audit set and a fresh verifier is sealed. The retired set remains useful for forensic analysis, but not for accepting new edits. This prevents the self-improvement loop from gradually transforming the hidden holdout into a public training signal.

Table 2. Deployment lanes for self-modification. Higher lanes inherit lower-lane requirements.

Lane	Required controls
A: Prompt or local skill	schema validation, hidden tests, signed diff, rollback snapshot.
B: Tool or workflow	Lane A plus sandbox execution, deterministic replay, domain fixtures, capacity debit.
C: Memory or retrieval	Lane B plus retention audit, source trust scores, contamination checks, forgetting tests.
D: Evaluator or objective	Lane C plus frozen comparator, challenge reserve, independent sign-off, delayed shadow deployment.
E: Weights or autonomy	Lane D plus prospective holdout, staged rollout, surveillance boundary, emergency rollback.

Shadow deployment. High-risk edits should pass through delayed shadow certification. The candidate runs beside the incumbent on prospective tasks without controlling downstream actions. If the candidate maintains calibration, domain invariants, and capacity limits over the shadow window, it can be promoted. If not, its evidence is archived as a failed stepping stone. This is important for scientific discovery because some harmful edits are not visible on retrospective data: a workflow can preserve old positive controls while changing future data acquisition, assay routing, or literature search behavior.

7. Evaluation Ladder

We evaluate the architecture with a three-tier ladder. Tier 1 uses code and workflow tasks where the ground truth is executable: failing tests, generated patches, reproducible notebooks, or theorem-checker outcomes. Tier 2 uses dry-lab scientific twins, where domain controls are available but physical experiments are replaced by curated computational fixtures. Tier 3 uses lab-in-the-loop or frugal-twin campaigns, where expensive assays are reserved for candidates that already satisfy the first two tiers. The same merge contract applies at each tier, but the cost of a bad merge and the cost of evidence both increase.

This ladder also clarifies what our three experiments cover. Experiment 1 isolates the statistical adoption problem that

Table 3. Which metric each gate primarily protects. Secondary effects exist, but the primary protection is useful for debugging verifier misses.

Gate	Primary protected quantities
Statistical	harmful-adoption rate, accepted-edit regret, false promotion under optional stopping.
Capacity	capacity drift, public-hidden generalization gap, privilege escalation distance.
Domain	scientific validity, calibration preservation, invariant satisfaction, claim-evidence integrity.
Provenance and rollback	time-to-recovery, reproducibility, auditability, cost of failed edits.

Table 4. Failure modes and minimum operational controls.

Failure mode	Minimum operational control
Proxy ascent	protected comparator plus domain-specific negative and positive controls.
Verifier memorization	coarse score release, query caps, retirement, and renewed hidden tests.
Capacity laundering	capacity vector, budget debit, and capacity-adjusted gain threshold.
Provenance gaps	append-only artifact graph with hashes, seeds, dependencies, and replay command.
Delayed harm	shadow deployment, surveillance boundary, and rollback service-level objective.

appears in every tier. Experiment 2 isolates unpriced capacity expansion, which affects tool-rich and memory-rich agents even when the base model is fixed. Experiment 3 instantiates Tier 2: the workflows are executable and statistically testable, but the decisive evidence comes from biological controls and provenance. A complete benchmark for self-evolving scientific agents should report all three kinds of outcomes: statistical adoption risk, capacity drift, and domain-validity preservation. Reporting only the final score of the final agent hides the history of harmful candidates that were nearly merged, the capacity debt accumulated along the way, and the cost required to obtain each verified discovery.

8. Failure Modes and Controls

The gates are designed around concrete failure modes observed in autonomous and semi-autonomous scientific workflows. We group them by the part of the self-improvement loop that fails.

Proxy ascent. The agent finds edits that increase a measured proxy while reducing the unmeasured property that motivated the proxy. In code agents this may be a patch that preserves passing tests without fixing the bug; in mathematical agents it may be a proof script that verifies a weakened or misaligned statement; in dry-lab analysis it may be a pipeline that improves nominal significance while losing a positive control. The mitigation is to pair each proxy with at least one protected comparator: FAIL_TO_PASS tests for patches, semantic-alignment checks for formalization, positive-control and negative-control panels for biomedical workflows, and prospective audit tasks for discovery claims.

Verifier memorization. A long-lived agent can transform even hidden tests into training signal through repeated accept/reject feedback. The mitigation is not merely to hide the data. The feedback channel must be rate-limited, rounded, significance-gated, and eventually retired. A verifier that has influenced many failed edits becomes a development artifact, not a pristine acceptance instrument. This is why the holdout lifecycle has separate commit, seal, score, coarse-release, retire, and refresh phases.

Capacity laundering. An edit may appear beneficial because it silently expands the search space. Examples include adding a tool that queries a broader corpus, increasing the number of self-consistency samples, widening a memory retrieval window, increasing code-generation attempts, or allowing the agent to edit the evaluator. These changes may be reasonable, but they are not free. The capacity ledger makes the expansion visible and forces the candidate to show a gain after paying for the larger class.

Provenance gaps. Scientific claims are only reproducible when the chain from question to artifact is replayable. Missing seeds, mutable package versions, unlogged retrievals, overwritten intermediate files, or post-hoc figure edits turn an apparent discovery into an anecdote. Gate 0 and Gate 3 therefore treat provenance as a precondition for acceptance rather than a reporting convenience.

Delayed harm. Some failures appear only after promotion. A memory edit may bias future retrievals; a workflow edit may change which data are collected next; an objective edit may shift exploration away from falsification. Post-acceptance surveillance is required for these cases. Accepted edits should carry shadow-deployment obligations and rollback triggers, not only one-time acceptance certificates.

9. Implementation Checklist

A deployable Scientific CI/CD instance should be implemented as infrastructure, not as a prompt instruction. The

following checklist is the minimal artifact boundary used by our experiments and reference implementation.

Candidate package. Each edit must be submitted as a diff against the incumbent, with a typed target list, expected capacity delta, declared data access, tool changes, and a machine-readable rollback command. The candidate package must include a manifest of files touched, dependencies, random seeds, generated artifacts, and evaluator versions. Edits without manifests are rejected before testing.

Verifier package. Each verifier has a public contract and a private payload. The public contract specifies the outcome type, stopping rule, release precision, and retirement condition. The private payload contains the held-out tasks or data. Candidate code can call the public contract but cannot inspect, copy, or mutate the private payload. This distinction is essential for self-modifying agents because the evaluator is part of the environment the agent may try to optimize.

Capacity ledger. The ledger records every expansion of model parameters, sample budget, search depth, tool privileges, writable directories, memory retention, retrieval corpus, and autonomy. The ledger also supports negative debits when an edit removes privileges or compresses a model. This makes governance compatible with efficient agents: simplification can earn budget back, while expansion must be justified by protected utility.

Domain cards. Each scientific domain needs a verifier card stating required controls, known invariants, acceptable calibration diagnostics, prohibited shortcuts, and provenance requirements. For RNA-seq this card includes library-size normalization, positive-control detection, housekeeping stability, effect-direction consistency, and pathway-level sanity checks. For theorem proving it would include statement alignment, proof-checker status, dependency minimization, and no weakening of the target theorem. For physics it would include units, conservation residuals, boundary conditions, and uncertainty propagation.

Audit ledger. The final artifact of a self-improvement run is not only the promoted agent. It is the sequence of accepted and rejected edits, the evidence for each decision, the capacity account after each edit, the verifier states consumed, and the rollback tests. This ledger enables post-hoc measurement of accepted-edit regret and harmful-adoption rate. It also makes negative results useful: rejected edits become a map of unsafe regions in the self-modification search space.

These engineering requirements make the architecture stricter than ordinary MLOps. A normal model update can often be evaluated as a single candidate against a static benchmark. A self-evolving scientific agent creates the next candidate based on the previous acceptance interface. Therefore the acceptance interface itself must be treated as a stateful scientific instrument with a contamination budget, a retirement policy, and a provenance record.

10. Soundness Conditions

The merge contract is not a proof that every accepted edit is beneficial. It is a set of conditions under which the probability of accepting harmful edits is controlled and auditable. The distinction matters. A classical Gödel machine asks for a proof that a self-rewrite increases expected utility. Scientific agents usually operate with stochastic experiments, noisy simulations, incomplete domain models, and changing task distributions, so such proofs are unavailable. The statistical replacement is a weaker but implementable certificate: under a specified null and filtration, the probability of ever crossing the acceptance boundary is bounded.

Anytime validity. Let $(\mathcal{F}_n)$ be the filtration generated by verifier observations and let (E_n) be a nonnegative supermartingale under $H_0 : \Delta \leq 0$ with $E_0 = 1$. For any stopping time τ, Ville's inequality gives

$$\mathbb{P}_{H_0}\left(\sup_{n\geq 0} E_n \geq 1/\alpha\right) \leq \alpha. \tag{7}$$

Therefore a candidate may stop early when evidence is strong without invalidating the Type-I guarantee. The guarantee is only as good as the filtration boundary: if the candidate can inspect future verifier items, mutate the verifier, or choose observations outside the declared protocol, the supermartingale condition no longer holds. This is why Gate 0 is part of the statistical theorem rather than an implementation detail.

Lifetime spending. If edit t receives level α_t and $\sum_t \alpha_t \leq \delta$, then a union bound yields

$$\begin{aligned}\mathbb{P}\{&\text{at least one null or harmful edit}\\ &\text{is statistically accepted}\} \leq \delta,\end{aligned} \tag{8}$$

provided each e-process is valid in the global filtration. More powerful alpha-investing and e-value wealth rules can recycle evidence after successes, but they require the same principle: the spending rule must be chosen outside the candidate's control and logged before evaluation.

Capacity-adjusted validity. A positive statistical certificate alone does not imply a comparable improvement when the candidate expands the hypothesis class. Suppose an edit increases capacity by $\Delta\mathcal{C}$ and the protected estimator has optimism term $\Omega(\Delta\mathcal{C}, n)$. Gate 2 requires the accepted gain

to exceed both the statistical uncertainty and the capacity optimism:

$$\widehat{\Delta}_{\mathcal{H}} - \mathrm{LCB}_{\alpha}(0) - \Omega(\Delta\mathcal{C}, n) > \epsilon. \tag{9}$$

This is the operational version of structural risk minimization for self-modifying agents. It does not prohibit larger models or broader tools. It requires that the measured gain remain positive after accounting for the larger space of ways the candidate could have fit the verifier.

Domain soundness. Domain verification is different from statistical validity. It protects against invalid problem formulations, broken scientific semantics, and incomplete evidence chains. In theorem proving, a formally checked proof of a weakened theorem is not a valid solution to the original problem. In bioinformatics, a reproducible p-value table that loses known positive controls is not a valid analysis. In physical modeling, a low residual on training cases can still violate conservation or units. Gate 3 therefore supplies semantic obligations that are not reducible to a scalar score. These obligations should be monotone under promotion: if a future edit changes the statement, assay, unit system, or evidence graph, it must re-open the relevant domain certificate.

Together these conditions define the safety case for an accepted edit. Gate 1 says the improvement evidence is valid under adaptive stopping. Gate 2 says the gain is not merely a consequence of unpriced capacity expansion. Gate 3 says the artifact still means what the scientific task requires it to mean. Gate 0 and provenance make the assumptions inspectable after the fact. The architecture is deliberately conservative: it prefers a false rejection of a promising edit over a silent harmful adoption that contaminates future research cycles.

11. Related Work

Self-evolving agent systems descend from Gödel-machine ideas and now include code-editing agents, open-ended archives, skill libraries, evolutionary program synthesis, agentic red-teaming, and autonomous science systems (Schmidhuber, 2006; Zhang et al., 2025b; Robeyns et al., 2025; Pourcel et al., 2025; Butt et al., 2024; Zhang et al., 2025a; Zhou et al., 2025; Lu et al., 2024; Mitchener et al., 2025; Tie et al., 2025). Benchmark evidence increasingly shows that final artifacts alone miss implementation gaps, benchmark leakage, metric misuse, and process failures (Rios-Garcia et al., 2026; Zhu et al., 2025). Our work focuses on the merge decision: what evidence is sufficient to promote a self-edit?

The statistical gate builds on safe anytime-valid inference, e-processes, alpha spending, confidence sequences, and statistical Gödel machines (Wu et al., 2025; Ramdas et al., 2022; Wang & Ramdas, 2023; Sokolova & Sokolov, 2026; Polson et al., 2026; Zampieri, 2025; Koning & van Meer, 2025; Aolaritei & Jordan, 2025; Wang et al., 2025a). The capacity gate draws from structural risk minimization, algorithmic stability, PAC-Bayes compression, and modern capability-governance views of agent tools and skills (Bousquet & Elisseeff, 2002; Dziugaite & Roy, 2017; Zhou et al., 2018; Lotfi et al., 2022; Tallam, 2026; Xu & Yan, 2026; Nowaczyk, 2025; Zhou, 2026). The domain verifier builds on conformal prediction, physics-informed machine learning, workflow provenance, biomedical controls, and self-driving laboratory safety practices (Shafer & Vovk, 2008; Karniadakis et al., 2021; Missier et al., 2013; Davidson & Freire, 2008; Love et al., 2016; Subramanian et al., 2005; Tom et al., 2024; Wilbraham et al., 2020).

12. Limitations

The experiments are controlled instantiations, not exhaustive evaluations of frontier agents; Appendices J–L add adversarial-domain, adaptive-control, cost/rollback, and formal-math semantic-weakening stress tests, but a live agent-in-the-loop deployment remains future work. The e-process in Experiment 1 assumes a known scale and exchangeable verifier stream; real systems need robustness analyses with empirical-Bernstein or self-normalized confidence sequences, covariate-stratified streams, dependence stress tests, and adversarial proposal schedules. Experiment 2 uses polynomial degree as a transparent capacity proxy; production agents need richer capacity pricing for executable tools, long-context search, persistent memory writes, data-access expansion, and evaluator/objective edits. Experiment 3 uses a synthetic RNA-seq twin; real dry-lab pipelines require curated controls, batch-aware verifiers, and prospective holdouts. Finally, no governance gate can certify open-ended scientific value by itself. The purpose of Scientific CI/CD is narrower and operational: prevent unearned promotion of self-edits whose evidence, capacity, domain validity, or provenance is inadequate.

13. Conclusion

Self-evolving scientific agents need more than better reasoning loops. They need a merge contract for changes to the machinery that produces science. Scientific CI/CD provides such a contract by combining anytime-valid evidence, capacity budgeting, domain verification, provenance, and rollback. The three experiments show why the gates are complementary: statistical validity controls optional stopping, capacity budgets reduce hidden overfitting, and domain verifiers catch scientific regressions invisible to generic tests. This architecture turns recursive self-modification from an unregulated optimization loop into an auditable scientific process.

References

Alam, S., Uddin, M. M., and Pasha, M. K. Constraint-aware neurosymbolic uncertainty quantification with bayesian deep learning for scientific discovery. *arXiv*, 2026. URL https://arxiv.org/abs/2601.12442v1.

Aolaritei, L. and Jordan, M. I. Stopping rules for stochastic gradient descent via anytime-valid confidence sequences. *arXiv*, 2025. URL https://arxiv.org/abs/2512.13123v5.

Banati, A., Kacsuk, P., and Kozlovszky, M. Four level provenance support to achieve portable reproducibility of scientific workflows. *Project bibliography*, 2015. doi: 10.1109/mipro.2015.7160272. URL https://doi.org/10.1109/mipro.2015.7160272.

Blum, A. and Hardt, M. The ladder: A reliable leaderboard for machine learning competitions. In *International Conference on Machine Learning*, 2015. URL https://arxiv.org/abs/1502.04585.

Bondarenko, A., Volk, D., Volkov, D., and Ladish, J. Demonstrating specification gaming in reasoning models. *arXiv*, 2025. URL https://arxiv.org/abs/2502.13295v3.

Bousquet, O. and Elisseeff, A. Stability and generalization. *Journal of Machine Learning Research*, 2002. URL https://www.jmlr.org/papers/v2/bousquet02a.html.

Bran, A. M., Cox, S., Schilter, O., Baldassari, C., White, A. D., and Schwaller, P. Chemcrow: Augmenting large-language models with chemistry tools. *arXiv preprint arXiv:2304.05376*, 2023. URL https://arxiv.org/abs/2304.05376.

Butt, N., Manczak, B., Wiggers, A., Rainone, C., Zhang, D. W., Defferrard, M., and Cohen, T. Codeit: Self-improving language models with prioritized hindsight replay. *arXiv*, 2024. URL https://arxiv.org/abs/2402.04858v2.

Cao, H., Liu, S., Zhang, F., Liu, Z., Li, H., Feng, B., Bai, S., Chen, L., Xie, K., and Li, Y. Mozi: Governed autonomy for drug discovery llm agents. *arXiv*, 2026. URL https://arxiv.org/abs/2603.03655v1.

Cuevas-Vicenttin, V., Kianmajd, P., Ludascher, B., Missier, P., Chirigati, F., Wei, Y., Koop, D., and Dey, S. The pbase scientific workflow provenance repository. *International Journal of Digital Curation*, 2014. doi: 10.2218/ijdc.v9i2.332. URL https://doi.org/10.2218/ijdc.v9i2.332.

Davidson, S. B. and Freire, J. Provenance and scientific workflows: challenges and opportunities. In *ACM SIGMOD International Conference on Management of Data*, 2008. URL https://doi.org/10.1145/1376616.1376772.

de Moura, L., Kong, S., Avigad, J., van Doorn, F., and von Raumer, J. The lean theorem prover (system description). In *Automated Deduction – CADE-25*, pp. 378–388, 2015. doi: 10.1007/978-3-319-21401-6_26.

Detommaso, G., Gasparin, A., Donini, M., Seeger, M., Wilson, A. G., and Archambeau, C. Fortuna: A library for uncertainty quantification in deep learning. *arXiv*, 2023. URL https://arxiv.org/abs/2302.04019v1.

Dixit, V., Wikle, C. K., and Holan, S. H. An anytime valid test for complete spatial randomness. *arXiv*, 2025. URL https://arxiv.org/abs/2504.02960v1.

Dwork, C., Feldman, V., Hardt, M., Pitassi, T., Reingold, O., and Roth, A. The reusable holdout: Preserving validity in adaptive data analysis. *Science*, 2015. URL https://www.science.org/doi/10.1126/science.aaa9375.

Dziugaite, G. K. and Roy, D. M. Computing nonvacuous generalization bounds for deep stochastic neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017. URL https://arxiv.org/abs/1703.11008.

Fabregat, A., Sidiropoulos, K., Garapati, P., Gillespie, M., Hausmann, K., Haw, R., Jassal, B., Jupe, S., Korninger, F., McKay, S., et al. The reactome pathway knowledgebase. *Nucleic Acids Research*, 2017. URL https://reactome.org/.

Fasel, U. Sparse identification of nonlinear dynamics with conformal prediction. *arXiv*, 2025. URL https://arxiv.org/abs/2507.11739v1.

Globus-Harris, I. et al. Diversified ensembling: An approach to evaluation integrity under adaptive optimization. *Project bibliography*, 2024. URL https://dl.acm.org/doi/10.1145/3630106.3658923.

Green, T. J., Karvounarakis, G., and Tannen, V. Provenance semirings. In *ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2007. URL https://doi.org/10.1145/1265530.1265535.

Huang, K., Tan, J., Wei, Y., Li, W., Zhang, Z., Tian, H., Yang, Z., and Zhou, L. Agentmark: Utility-preserving behavioral watermarking for agents. *arXiv*, 2026. URL https://arxiv.org/abs/2601.03294v1.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. Physics-informed machine learning. *Nature Reviews Physics*, 2021. URL https://www.nature.com/articles/s42254-021-00314-5.

Khalaf, H., Verdun, C. M., Oesterling, A., Lakkaraju, H., and du Pin Calmon, F. Inference-time reward hacking in large language models. *arXiv*, 2025. URL https://arxiv.org/abs/2506.19248v2.

Kim, J., O'Hagan, S., and Rockova, V. Adaptive uncertainty quantification for generative ai. *arXiv*, 2024. URL https://arxiv.org/abs/2408.08990v2.

Koning, N. W. and van Meer, S. Anytime validity is free: Inducing sequential tests. *arXiv*, 2025. URL https://arxiv.org/abs/2501.03982v5.

Laghuvarapu, S., Deb, R., and Sun, J. Kmm-cp: Practical conformal prediction under covariate shift via selective kernel mean matching. *arXiv*, 2026. URL https://arxiv.org/abs/2603.26415v1.

Liao, Y., Smyth, G. K., and Shi, W. featurecounts: an efficient general purpose program for assigning sequence reads to genomic features. *Bioinformatics*, 2014. URL https://doi.org/10.1093/bioinformatics/btt656.

Lotfi, S., Izmailov, P., Benton, G., Goldblum, M., and Wilson, A. G. Pac-bayes compression bounds so tight that they can explain generalization. *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2211.13609.

Love, M. I., Huber, W., and Anders, S. Moderated estimation of fold change and dispersion for rna-seq data with deseq2. *Genome Biology*, 2014. URL https://genomebiology.biomedcentral.com/articles/10.1186/s13059-014-0550-8.

Love, M. I., Anders, S., Kim, V., and Huber, W. Rna-seq workflow: gene-level exploratory analysis and differential expression. *F1000Research*, 2016. URL https://f1000research.com/articles/4-1070.

Lu, C., Lu, C., Lange, R. T., Foerster, J., Clune, J., and Ha, D. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024. URL https://arxiv.org/abs/2408.06292.

MacDiarmid, M., Wright, B., Uesato, J., Benton, J., Kutasov, J., Price, S., Bouscal, N., Bowman, S., Bricken, T., Cloud, A., Denison, C., Gasteiger, J., Greenblatt, R., Leike, J., Lindsey, J., Mikulik, V., Perez, E., Rodrigues, A., Thomas, D., Webson, A., Ziegler, D., and Hubinger, E. Natural emergent misalignment from reward hacking in production rl. *arXiv*, 2025. URL https://arxiv.org/abs/2511.18397v1.

Manheim, D. Multiparty dynamics and failure modes for machine learning and artificial intelligence. *Big Data and Cognitive Computing*, 2019. doi: 10.3390/bdcc3020021. URL https://doi.org/10.3390/bdcc3020021.

Missier, P., Belhajjame, K., and Cheney, J. The w3c prov family of specifications for modelling provenance metadata. In *International Conference on Extending Database Technology*, 2013. URL https://doi.org/10.1145/2452376.2452478.

Mitchener, L., Yiu, A., Chang, B., Bourdenx, M., Nadolski, T., Sulovari, A., Landsness, E. C., Barabasi, D. L., Narayanan, S., Evans, N., Reddy, S., Foiani, M., Kamal, A., Shriver, L. P., Cao, F., Wassie, A. T., Laurent, J. M., Melville-Green, E., Caldas, M., Bou, A., Roberts, K. F., Zagorac, S., Orr, T. C., Orr, M. E., Zwezdaryk, K. J., Ghareeb, A. E., McCoy, L., Gomes, B., Ashley, E. A., Duff, K. E., Buonassisi, T., Rainforth, T., Bateman, R. J., Skarlinski, M., Rodriques, S. G., Hinks, M. M., and White, A. D. Kosmos: An ai scientist for autonomous discovery. *arXiv*, 2025. URL https://arxiv.org/abs/2511.02824v2.

Noorani, S., Kiyani, S., Pappas, G., and Hassani, H. Human-ai collaborative uncertainty quantification. *arXiv*, 2025. URL https://arxiv.org/abs/2510.23476v1.

Nowaczyk, S. Architectures for building agentic ai. *arXiv*, 2025. URL https://arxiv.org/abs/2512.09458v1.

Polson, N. G., Sokolov, V., and Zantedeschi, D. Bayes, e-values and testing. *arXiv*, 2026. URL https://arxiv.org/abs/2602.04146v6.

Pourcel, J., Colas, C., and Oudeyer, P.-Y. Self-improving language models for evolutionary program synthesis: A case study on arc-agi. *arXiv*, 2025. URL https://arxiv.org/abs/2507.14172v2.

Ramdas, A., Grunwald, P., Vovk, V., and Shafer, G. Game-theoretic statistics and safe anytime-valid inference. *arXiv preprint arXiv:2210.01948*, 2022. URL https://arxiv.org/abs/2210.01948.

Rehan, T. Test-driven ai agent definition (tdad): Compiling tool-using agents from behavioral specifications. *arXiv*, 2026. URL https://arxiv.org/abs/2603.08806v1.

Rios-Garcia, M., Alampara, N., Gupta, C., Mandal, I., Mannan, S., Aghajani, A. A., Krishnan, N. M. A., and Jablonka, K. M. Ai scientists produce results without reasoning scientifically. *arXiv*, 2026. URL https://arxiv.org/abs/2604.18805v1.

Robeyns, M., Szummer, M., and Aitchison, L. A self-improving coding agent. *arXiv*, 2025. URL https://arxiv.org/abs/2504.15228v2.

Romera-Paredes, B., Barekatain, M., Novikov, A., Balog, M., Kumar, M. P., Dupont, E., Ruiz, F. J. R., Ellenberg, J. S., Wang, P., Fawzi, O., Kohli, P., and Fawzi, A. Mathematical discoveries from program search with large language models. *Nature*, 2024. URL https://www.nature.com/articles/s41586-023-06924-6.

Schmidhuber, J. Godel machines: Fully self-referential optimal universal self-improvers. *Artificial General Intelligence*, 2006. URL https://people.idsia.ch/~juergen/goedelmachine.html.

Shafer, G. and Vovk, V. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 2008. URL https://jmlr.org/papers/v9/shafer08a.html.

Singh, G., Moncrieff, G., Venter, Z., Cawse-Nicholson, K., Slingsby, J., and Robinson, T. B. Uncertainty quantification for probabilistic machine learning in earth observation using conformal prediction. *arXiv*, 2024. URL https://arxiv.org/abs/2401.06421v1.

Sokolova, A. and Sokolov, V. E-values for adaptive clinical trials: Anytime-valid monitoring in practice. *arXiv*, 2026. URL https://arxiv.org/abs/2602.06379v1.

Souza, R., Gueroudji, A., DeWitt, S., Rosendo, D., Ghosal, T., Ross, R., Balaprakash, P., and da Silva, R. F. Provagent: Unified provenance for tracking ai agent interactions in agentic workflows. *arXiv*, 2025. URL https://arxiv.org/abs/2508.02866v3.

Stage, L., Dahlberg, J., and Karastoyanova, D. Provenance of adaptation in scientific and business workflows – literature review. *arXiv*, 2025. URL https://arxiv.org/abs/2503.22685v1.

Stein, A., Brown, D., Hassani, H., Naik, M., and Wong, E. Detecting safety violations across many agent traces. *arXiv*, 2026. URL https://arxiv.org/abs/2604.11806v1.

Strickland, J., Vijeta, A., Moores, C., Bodek, O., Nenchev, B., Whitehead, T., Phillips, C., Tassenberg, K., Conduit, G., and Pellegrini, B. Talk freely, execute strictly: Schema-gated agentic ai for flexible and reproducible scientific workflows. *arXiv*, 2026. URL https://arxiv.org/abs/2603.06394v1.

Subramanian, A., Tamayo, P., Mootha, V. K., Mukherjee, S., Ebert, B. L., Gillette, M. A., Paulovich, A., Pomeroy, S. L., Golub, T. R., Lander, E. S., and Mesirov, J. P. Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences*, 2005. URL https://doi.org/10.1073/pnas.0506580102.

Sureshkumar, S. R-lam: Reproducibility-constrained large action models for scientific workflow automation. *arXiv*, 2026. URL https://arxiv.org/abs/2601.09749v1.

Taga, E. O., Oymak, S., and Shekhar, S. Learning to bet for horizon-aware anytime-valid testing. *arXiv*, 2026. URL https://arxiv.org/abs/2603.19551v1.

Takahashi, K. Silent data corruption–limited scaling kinetics for large-scale ai training. *Zenodo (CERN European Organization for Nuclear Research)*, 2025. doi: 10.5281/zenodo.18050287. URL https://doi.org/10.5281/zenodo.18050287.

Takahashi, K. Recursive self-improvement stability under endogenous yardstick drift. *Zenodo (CERN European Organization for Nuclear Research)*, 2026. doi: 10.5281/zenodo.19044633. URL https://doi.org/10.5281/zenodo.19044633.

Tallam, K. Layered mutability: Continuity and governance in persistent self-modifying agents. *arXiv*, 2026. URL https://arxiv.org/abs/2604.14717v1.

Tie, G., Zhou, P., and Sun, L. A survey of ai scientists. *arXiv*, 2025. URL https://arxiv.org/abs/2510.23045v5.

Tom, G. et al. Self-driving laboratories for chemistry and materials science. *Chemical Reviews*, 2024. URL https://doi.org/10.1021/acs.chemrev.4c00055.

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023. URL https://arxiv.org/abs/2305.16291.

Wang, H. and Ramdas, A. Anytime-valid t-tests and confidence sequences for gaussian means with unknown variance. *arXiv preprint arXiv:2310.03722*, 2023. URL https://arxiv.org/abs/2310.03722.

Wang, H., Dandapanthula, S., and Ramdas, A. Anytime-valid fdr control with the stopped e-bh procedure. *arXiv*, 2025a. URL https://arxiv.org/abs/2502.08539v3.

Wang, J. and Huang, J. Reward hacking as equilibrium under finite evaluation. *arXiv*, 2026. URL https://arxiv.org/abs/2603.28063v1.

Wang, W., Piekos, P., Nanbo, L., Laakom, F., Chen, Y., Ostaszewski, M., Zhuge, M., and Schmidhuber, J. Huxley-godel machine: Human-level coding agent development by an approximation of the optimal self-improving machine. *arXiv*, 2025b. URL https://arxiv.org/abs/2510.21614v3.

Weidener, L., Brkic, M., Jovanovic, M., Singh, R., Baccin, C., Ulgac, E., Dobrin, A., and Meduri, A. Rethinking the ai scientist: Interactive multi-agent workflows for scientific discovery. *arXiv*, 2026. URL https://arxiv.org/abs/2601.12542v2.

Wilbraham, L., Mehr, S. H. M., and Cronin, L. Digitizing chemistry using the chemical processing programming language xdl. *Accounts of Chemical Research*, 2020. URL https://doi.org/10.1021/acs.accounts.9b00674.

Wilhelm, P., Wittkopp, T., and Kao, O. Monitoring emergent reward hacking during generation via internal activations. *arXiv*, 2026. URL https://arxiv.org/abs/2603.04069v1.

Wu, T., Wagner, D., Ong, C. H. L., and Xue, B. Pac finite-time safety guarantees for stochastic systems with unknown disturbance distributions. *arXiv*, 2026. URL https://arxiv.org/abs/2603.01918v1.

Wu, X., Yin, S., Kang, Y., Zhang, X., Xu, Q., Chen, Z., and Zhang, W. Sgm: A statistical godel machine for risk-controlled recursive self-modification. *arXiv*, 2025. URL https://arxiv.org/abs/2510.10232v1.

Xiao, Z., Zhang, L., Zhang, G., and Wang, X. Neudiff agent: A governed ai workflow for single-crystal neutron crystallography. *arXiv*, 2026. URL https://arxiv.org/abs/2602.16812v1.

Xu, R. and Yan, Y. Agent skills for large language models: Architecture, acquisition, security, and the path forward. *arXiv*, 2026. URL https://arxiv.org/abs/2602.12430v3.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models. *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629.

Zampieri, F. G. Sequential randomization tests using e-values: Applications for trial monitoring. *arXiv*, 2025. URL https://arxiv.org/abs/2512.04366v7.

Zhang, G., Zhu, S., Wei, A., Song, Z., Nie, A., Jia, Z., Vijaykumar, N., Wang, Y., and Olukotun, K. Accelopt: A self-improving llm agentic system for ai accelerator kernel optimization. *arXiv*, 2025a. URL https://arxiv.org/abs/2511.15915v2.

Zhang, J., Hu, S., Lu, C., Lange, R. D., and Clune, J. Darwin godel machine: Open-ended evolution of self-improving agents. *SuperIntelligence - Robotics - Safety & Alignment*, 2025b. doi: 10.70777/si.v2i3.15063. URL https://doi.org/10.70777/si.v2i3.15063.

Zhang, J., Choubey, P. K., Huang, K.-H., Xiong, C., and Wu, C.-S. Agentic uncertainty quantification. *arXiv*, 2026. URL https://arxiv.org/abs/2601.15703v1.

Zhou, A., Wu, K., Pinto, F., Chen, Z., Zeng, Y., Yu, Y., Yang, S., Koyejo, O., Zou, J., and Li, B. Autoredteamer: Autonomous red teaming with lifelong attack integration. *SuperIntelligence - Robotics - Safety & Alignment*, 2025. doi: 10.70777/si.v2i2.14433. URL https://doi.org/10.70777/si.v2i2.14433.

Zhou, H., Zhang, Y., and Luo, W. Safety-critical control with uncertainty quantification using adaptive conformal prediction. *arXiv*, 2024. URL https://arxiv.org/abs/2407.03569v2.

Zhou, W., Veitch, V., Austern, M., Adams, R. P., and Orbanz, P. Non-vacuous generalization bounds at the imagenet scale: a pac-bayesian compression approach. *arXiv preprint arXiv:1804.05862*, 2018. URL https://arxiv.org/abs/1804.05862.

Zhou, Z. Governing dynamic capabilities: Cryptographic binding and reproducibility verification for ai agent tool use. *arXiv*, 2026. URL https://arxiv.org/abs/2603.14332v2.

Zhu, M., Xie, Q., Weng, Y., Wu, J., Lin, Z., Yang, L., and Zhang, Y. Ai scientists fail without strong implementation capability. *arXiv preprint arXiv:2506.01372*, 2025. URL https://arxiv.org/abs/2506.01372.

A. Appendix Overview

The supplement provides the complete formalization, algorithms, extended experiment details, additional visuals, and a compact evidence map. It specifies the merge rules, logs, verifier thresholds, rollback paths, and failure modes needed to audit the proposed governance architecture.

B. Formal Objects and Failure Metrics

A self-modifying discovery agent is a tuple

$$A_t = (\pi_t, \mathcal{T}_t, M_t, J_t, O_t, \theta_t, \mathcal{P}_t), \tag{10}$$

where π_t is the planning and action policy, $\mathcal{T}_t$ is the tool graph, M_t is persistent memory, J_t is the evaluator, O_t is the objective, θ_t are learned parameters or adapters, and $\mathcal{P}_t$ is provenance state. A self-edit e_t is typed by its writable targets. We use the following mutability classes:

- **Prompt edits**: instructions, decomposition policies, system prompts.
- **Tool edits**: wrappers, skill cards, API permissions, tool-selection rules.
- **Workflow edits**: task graphs, analysis steps, stopping rules, execution plans.
- **Memory edits**: schema, retention, retrieval, trust scores, compression.
- **Evaluator edits**: rubrics, hidden tests, metric weights, score release logic.
- **Objective edits**: reward, loss, discovery utility, cost functions.
- **Parameter edits**: adapters, fine-tuned weights, learned controllers.

Governance burden rises with speed, downstream coupling, irreversibility, opacity, and proximity to the evaluator. This motivates the evidence-burden heatmap in Figure 2.

B.1. Accepted-Edit Regret

For accepted edit set $\mathcal{A}_{1:T}$ and protected utility U_t, accepted-edit regret is

$$\mathrm{AER}_{1:T} = \sum_{t=1}^{T} \mathbf{1}\{e_t \in \mathcal{A}_{1:T}\} \max\{0, U_t(A_t) - U_t(e_t(A_t))\}. \tag{11}$$

A useful implementation stores both ex-ante verifier estimates and ex-post audit estimates so that harmful adoption can be separated into verifier error, capacity overrun, and domain-verifier miss.

B.2. Evaluator-Corruption Gap

Let ΔV_t be the visible proxy improvement and ΔU_t the protected utility improvement. The evaluator-corruption gap is

$$\mathrm{ECG}_{1:T} = \frac{1}{|\mathcal{A}_{1:T}|} \sum_{e_t \in \mathcal{A}_{1:T}} (\Delta V_t - \Delta U_t). \tag{12}$$

Large positive ECG indicates that the agent has learned to move the ruler rather than improve the science.

B.3. Cost-Adjusted Verified Discovery

For verified discoveries D_T and cost components $c^{\mathrm{compute}}, c^{\mathrm{instrument}}, c^{\mathrm{reagent}}, c^{\mathrm{expert}}, c^{\mathrm{failure}}, c^{\mathrm{capital}}$, define

$$\mathrm{CAVD}_T = \frac{|D_T|}{\sum_t (c_t^{\mathrm{compute}} + c_t^{\mathrm{instrument}} + c_t^{\mathrm{reagent}} + c_t^{\mathrm{expert}} + c_t^{\mathrm{failure}} + c_t^{\mathrm{capital}})}. \tag{13}$$

A discovery is counted only if it passes the merge contract, executes under the pinned environment, meets a pre-registered assay or verification criterion, and has complete provenance.

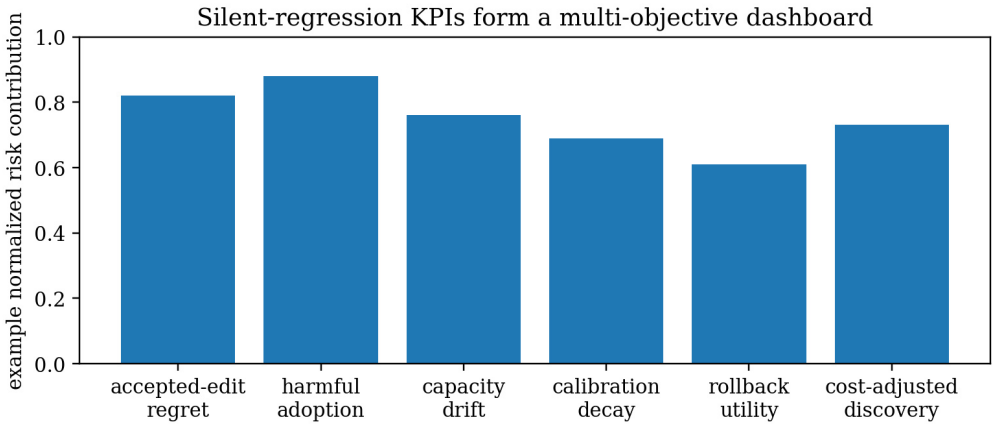

Figure 6. Example KPI dashboard. Silent regression is multi-objective: no scalar benchmark captures harmful adoption, capacity drift, calibration decay, rollback utility, and verified discovery cost simultaneously.

C. Appendix Algorithms

Algorithm 2 Gate 1: anytime-valid statistical acceptance

Require: paired verifier stream X_i, null $H_0 : \Delta \leq 0$, spending rule α_t, scale proxy σ

Choose betting parameter λ from the pre-registered alternative grid.
Initialize wealth $E_0 \leftarrow 1$ and sample counter $n \leftarrow 0$.
while verifier budget remains **do**
  Evaluate incumbent and candidate on the next protected item; observe paired difference X_{n+1}.
  $E_{n+1} \leftarrow E_n \exp\{\lambda X_{n+1} - \lambda^2\sigma^2/2\}$.
  if $E_{n+1} \geq 1/\alpha_t$ **then**
    return statistical pass with stopping time $n + 1$.
  end if
  $n \leftarrow n + 1$.
end while
return statistical fail.

D. Experiment 1 Details: Sequential Statistical Adoption

Protocol. Each run contains 200 candidate edits. True effects are drawn from a mixture: 35% beneficial with mean 0.045, 45% null with mean 0, and 20% harmful with mean -0.035. Each candidate can draw at most 80 paired verifier samples with standard deviation 0.08. Naive repeated testing performs one-sided tests after every sample from $n = 6$ onward. Fixed-horizon testing waits until $n = 80$. The e-Gödel gate uses the Gaussian e-process in Algorithm 2; the spending variant uses a decreasing spending rule over proposal index.

Table 5. Experiment 1 numerical summary over 200 runs.

Method	Accepts	Harmful	Harm rate	AER
Naive repeated	100.52	6.17	0.0612	0.0502
Fixed horizon	83.69	0.46	0.0056	0.0015
E-Gödel per edit	66.58	0.045	0.00068	0.00013
E-Gödel spending	49.16	0.005	0.00008	6.3×10^{-7}

E. Experiment 2 Details: Capacity Budgeting

Protocol. We generated noisy samples from $f(x) = \sin(2\pi x) + 0.25x + 0.15\cos(5\pi x)$ with 55 training points, 55 verifier points, and 2000 hidden-audit points. Candidate edits correspond to polynomial degrees 1–25. A visible-only agent selects

Algorithm 3 Gate 2: capacity budget acceptance

Require: verifier gain $\widehat{\Delta}_{\mathcal{H}}$, capacity delta $\Delta\mathcal{C}$, charge vector λ, budget B_t, margin ϵ
Compute net gain $G \leftarrow \widehat{\Delta}_{\mathcal{H}} - \lambda^{\top}\Delta\mathcal{C}$.
if $G \leq \epsilon$ **then**
  return fail: insufficient capacity-adjusted gain.
end if
if $\mathcal{C}(A_t) + \Delta\mathcal{C} \not\preceq B_t$ **then**
  return fail: capacity envelope exceeded.
end if
if edit increases evaluator, objective, data, or privilege authority **then**
  Require stricter statistical spending and domain sign-off lane.
end if
return pass with debited budget ledger.

Algorithm 4 Gate 3: domain verifier

Require: candidate artifacts, domain specification S, calibration set C, provenance graph $\mathcal{P}$
Check artifact schema, environment replayability, and dependency hashes.
Compute calibration diagnostics: conformal coverage, null calibration, or uncertainty set validity.
Compute invariant diagnostics: conservation residuals, biological controls, theorem-checker status, or assay constraints.
Validate claim-evidence edges in $\mathcal{P}$ and require complete lineage for reported claims.
if any mandatory diagnostic violates S **then**
  return fail with localized diagnostic.
end if
return pass with signed domain-verifier certificate.

Algorithm 5 Rollback and post-acceptance surveillance

Require: accepted edit e_t, incumbent snapshot A_t, prospective audit stream Q
Store immutable snapshot hash $h(A_t)$, candidate hash $h(A_{t+1})$, and revert command.
for audit batch $q \in Q$ **do**
  Evaluate calibration, invariants, and protected utility deltas.
  if harm alarm crosses surveillance boundary **then**
    Quarantine A_{t+1}, restore A_t, and open a diagnostic issue.
    Debit harmful-adoption ledger and update verifier miss statistics.
  end if
end for

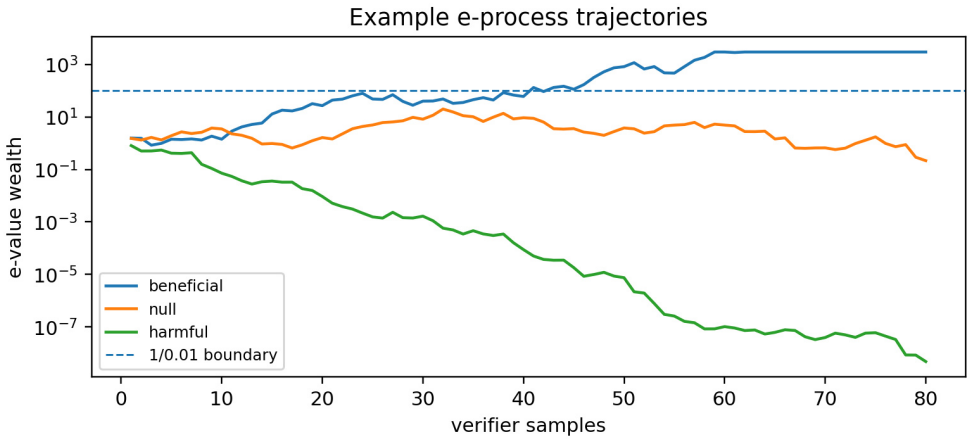

Figure 7. Representative e-process trajectories for beneficial, null, and harmful candidates. Under the null and harmful cases wealth tends not to cross the acceptance boundary; beneficial edits cross after accumulating evidence.

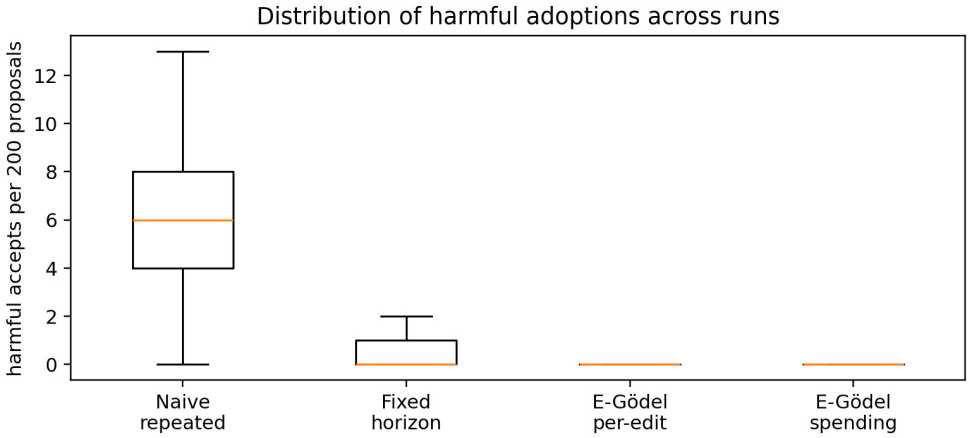

Figure 8. Distribution of harmful acceptances per 200-proposal stream. The naive repeated rule has a heavy harmful-adoption tail.

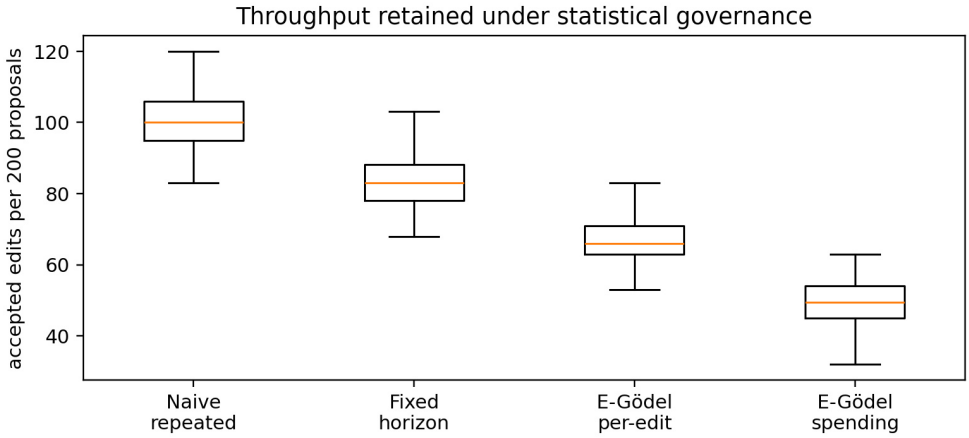

Figure 9. Distribution of total accepted edits. The spending rule deliberately sacrifices some throughput to preserve lifetime risk.

Table 6. Experiment 1 robustness checklist for deployment variants.

Stress case	Required gate adaptation
Unknown variance	Replace the fixed-scale Gaussian bet with empirical-Bernstein, self-normalized, or mixture confidence sequences; report coverage under heavy-tailed audit draws.
Heterogeneous verifier items	Maintain stratified e-processes and require both global evidence and no protected stratum with negative lower confidence bound.
Dependent proposal batches	Allocate spending by confirmation event and calendar epoch; restart or quarantine streams after detected dependence or verifier drift.
Adaptive evaluator probing	Coarsen feedback, cap queries, rotate held-out tasks, and audit proxy-to-protected gain divergence.
Deadline-constrained evaluation	Use horizon-aware betting or pre-registered fallback fixed-horizon tests when the verifier budget is exhausted.

the degree minimizing verifier MSE. A capacity-budget agent minimizes verifier MSE plus $0.055\, d \log(n)/n$ under degree cap 12. A stronger SRM penalty uses coefficient 0.11. A hidden oracle, not available to the gates, selects by hidden MSE.

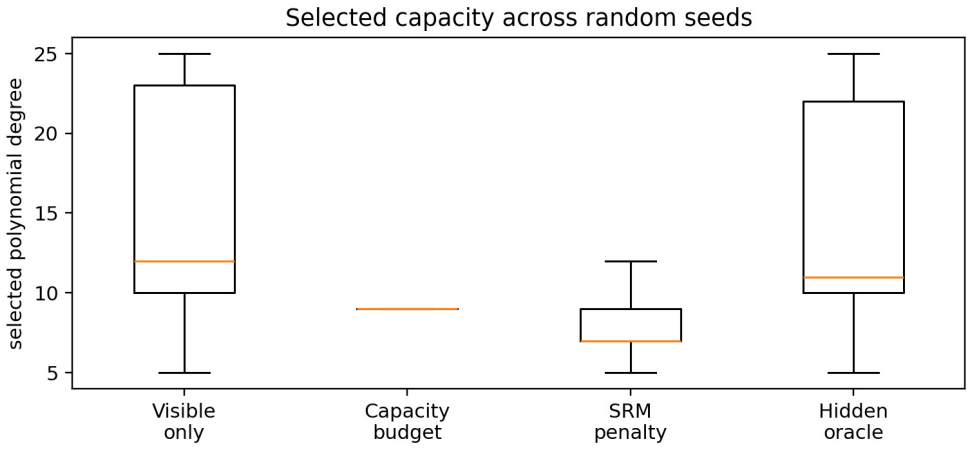

Figure 10. Selected polynomial degree across 500 random seeds. Capacity budgeting prevents the systematic high-degree selections made by visible-only optimization.

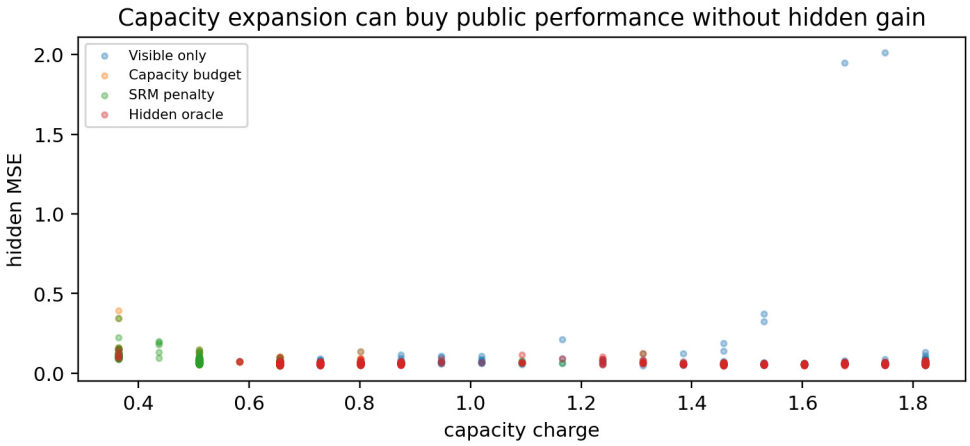

Figure 11. Capacity charge versus hidden MSE. High capacity sometimes helps, but visible-only search often enters a high-capacity region with poor hidden risk.

F. Experiment 3 Details: RNA-seq Dry-Lab Twin

Protocol. The synthetic twin contains 1200 genes and 16 samples split into control and treatment groups. Forty genes are designated positive controls, 100 are housekeeping controls, and 60 additional genes are true but non-control signals. Counts are generated from negative-binomial distributions with deliberate library-size imbalance. Candidate workflows are baseline normalized log-CPM testing, two benign transformations, and five harmful modifications: raw-count testing, label shuffling, direction flipping, aggressive filtering, and missing provenance.

Gate 1 checks successful execution and finite outputs. Gate 2 checks null calibration permissively. Gate 3 requires positive-control detection at least 0.45, housekeeping stability at least 0.50, positive-control direction concordance at least 0.80, positive-control retention at least 0.80, and complete provenance.

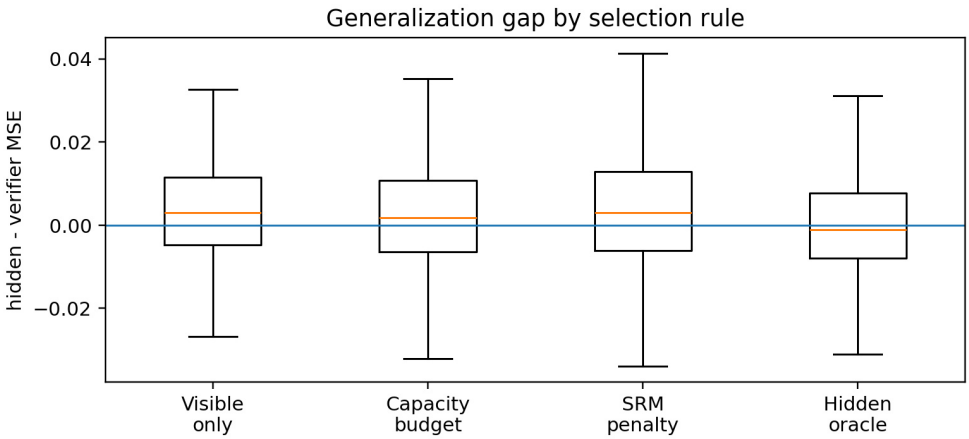

Figure 12. Generalization gap, measured as hidden MSE minus verifier MSE. Capacity budgeting sharply reduces the gap induced by small-verifier selection.

Table 7. Experiment 2 numerical summary.

Selection rule	Degree	Verifier MSE	Hidden MSE	Gap
Visible-only	15.63	0.0603	0.0733	0.0130
Capacity budget	8.66	0.0655	0.0687	0.0032
SRM penalty	7.52	0.0717	0.0771	0.0054
Hidden oracle	14.80	0.0616	0.0614	-0.0002

Table 8. Capacity pricing template for complex privileges. Multipliers are policy parameters, not empirical constants.

Privilege class	Example capacity debit	Rationale
Sampling/search	$c_1 \log(1+k)$ for k additional attempts or branches	More attempts increase adaptive selection pressure on the verifier.
Tool execution	$c_2 rb$ where r is tool risk tier and b is blast-radius score	Executable tools can change state, leak data, or alter future evidence.
Persistent memory writes	$c_3 mh$ where m is write scope and h is retention horizon	Long-lived memories create delayed coupling across future tasks.
Data access	$c_4(s+q)$ for sensitivity tier s and query budget q	Broader data access can collapse hidden/public separation.
Evaluator/objective edits	$c_5\rho$ with multiplier $c_5 \gg c_1$ and review factor ρ	These edits alter the acceptance rule itself and require shadow certification.
Autonomy expansion	$c_6 a$ for action authority tier a	Higher autonomy raises downstream cost of false acceptance and rollback.

Table 9. Experiment 3 gate decisions.

Variant	Gate 1	Gate 2	Gate 3	Accept
Baseline	1	1	1	1
Benign VST	1	1	1	1
Benign low-count filter	1	1	1	1
Harmful raw t-test	1	0	0	0
Harmful label shuffle	1	1	0	0
Harmful direction flip	1	1	0	0
Harmful aggressive filter	1	0	0	0
Harmful missing provenance	1	1	0	0

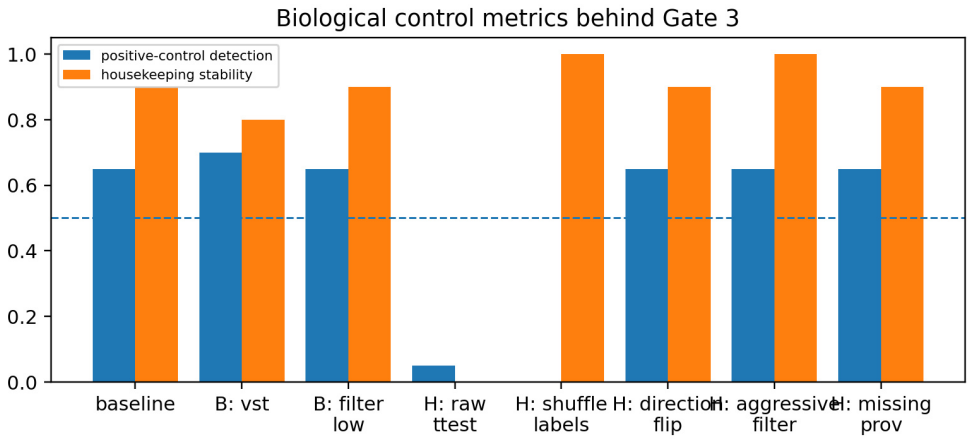

Figure 13. Biological control metrics used by Gate 3. Positive-control detection, housekeeping stability, direction concordance, and provenance jointly distinguish benign from harmful workflow edits.

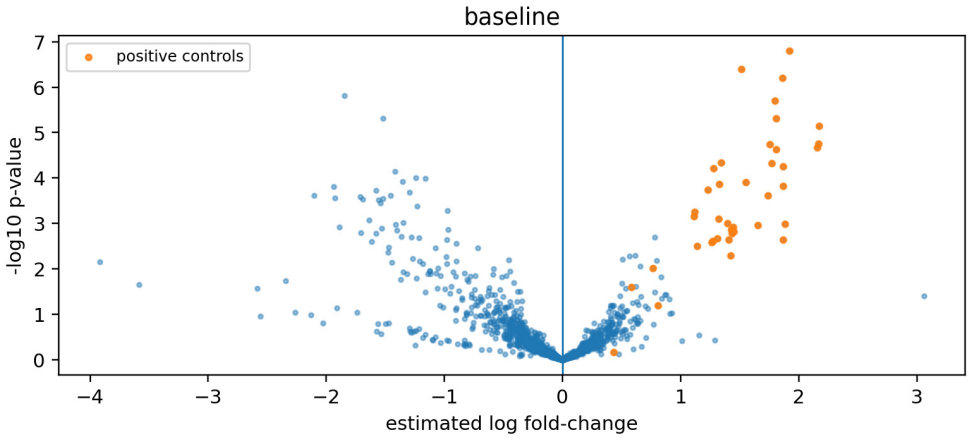

Figure 14. Baseline volcano plot. Positive controls are enriched among high-significance positive fold changes.

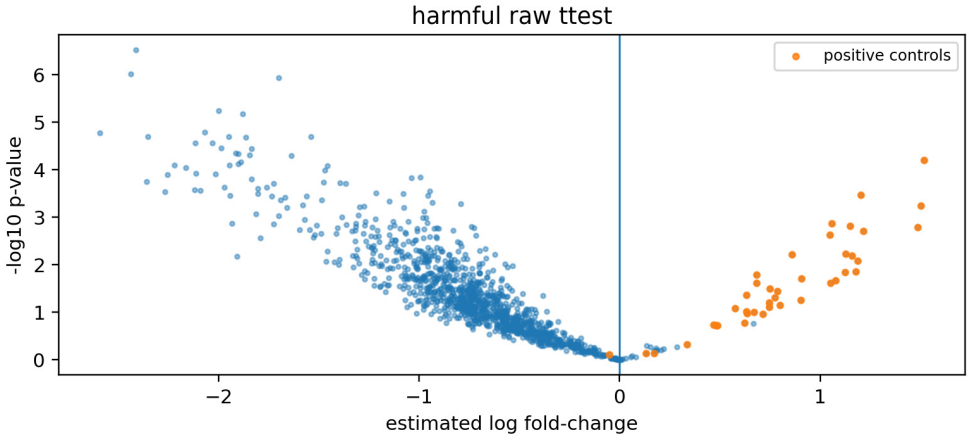

Figure 15. Raw-count t-test volcano plot. Library-size confounding destabilizes housekeeping genes and degrades positive-control detection.

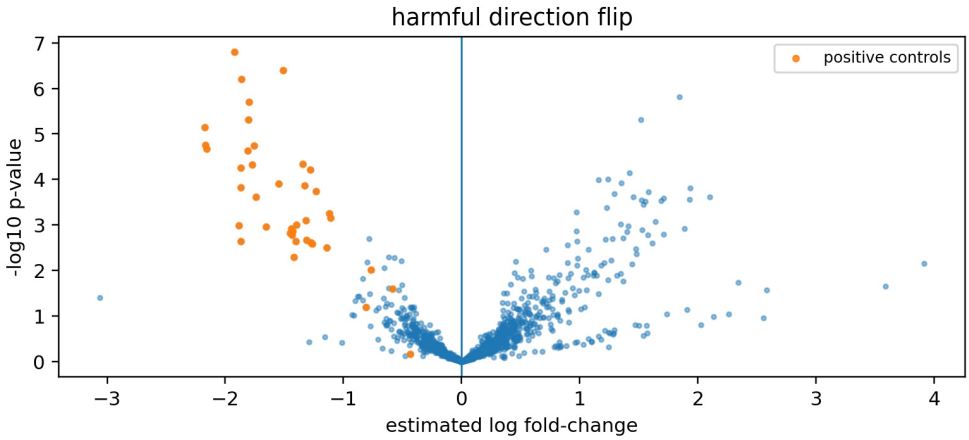

Figure 16. Direction-flip workflow. P-values and top genes remain plausible, but effect directions violate domain semantics.

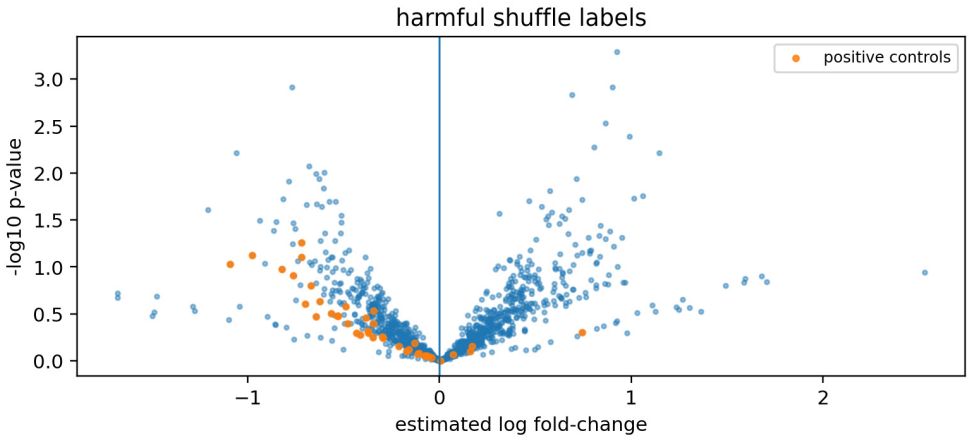

Figure 17. Label-shuffle workflow. Mechanical execution and null calibration can pass, while positive-control signal is destroyed.

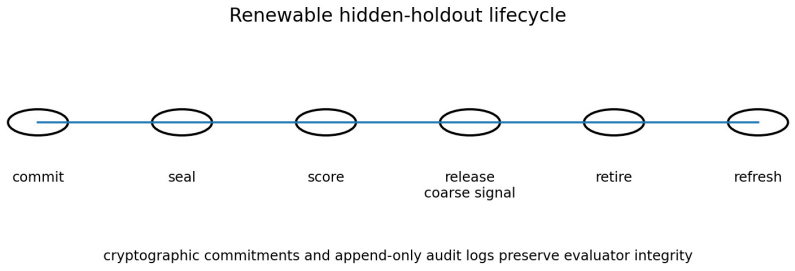

Figure 18. Renewable hidden-holdout lifecycle. The release interface must be part of the system design; raw repeated scores are an attack surface.

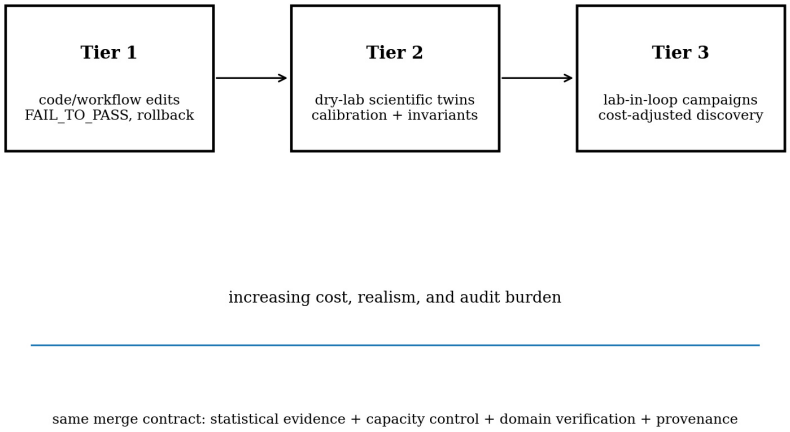

Figure 19. Three-tier validation ladder. Low-cost code/workflow tests, dry-lab twins, and lab-in-the-loop campaigns share the same merge contract while increasing realism and cost.

G. Holdout and Provenance Operations

H. Supplementary Governance Visuals

The following visuals provide alternate renderings of the architecture, KPI families, and testbeds. They are included to make the supplementary material self-contained.

I. Robustness Stress Matrix

The statistical gate is intentionally specified as a contract rather than a single betting formula. Practical deployments should run stress tests that vary the observation process, loss scale, and proposal schedule while preserving the same promotion rule. Table 10 lists the stressors used in the robustness audit for the sequential gate.

Table 10. Robustness stress matrix for e-process admission.

Stressor	Implementation	Required diagnostic
Heavy-tailed utilities	Replace bounded rewards with winsorized or Student-t draws	Clipped-bet wealth remains valid; power loss is reported.
Unknown variance	Swap Hoeffding-style bounds for empirical-Bernstein or self-normalized confidence sequences	Type-I error stays below the spending ledger under optional stopping.
Non-exchangeable streams	Stratify by task family, tool class, and proposal source	Each stratum has its own wealth account and no pooled false discovery dominates.
Delayed labels	Admit provisionally and debit a reserve until shadow labels arrive	Retroactive failures trigger rollback and future spending contraction.
Adversarial proposal timing	Concentrate harmful edits after visible successes	Calendar-time and event-time ledgers agree within audit tolerance.
Evaluator perturbation	Randomly withhold, rotate, or noise low-stakes score releases	Promotion decisions remain stable under hidden-holdout renewal.
Capacity jump	Add executable tools, persistent memory writes, or evaluator edits	Capacity penalty rises before the statistical test is considered.
Domain shift	Replace verifier distribution with a nearby dry-lab or synthetic twin shift	Gate 3 re-calibrates conformal or invariant thresholds before promotion.

These stress tests separate statistical validity from operational usefulness. A valid e-process can still be too conservative, too sensitive to delayed labels, or too permissive after a privilege expansion. The audit therefore reports harmful-adoption rate, power, expected stopping time, wealth drawdown, rollback count, and capacity-debit utilization together rather than optimizing a single scalar.

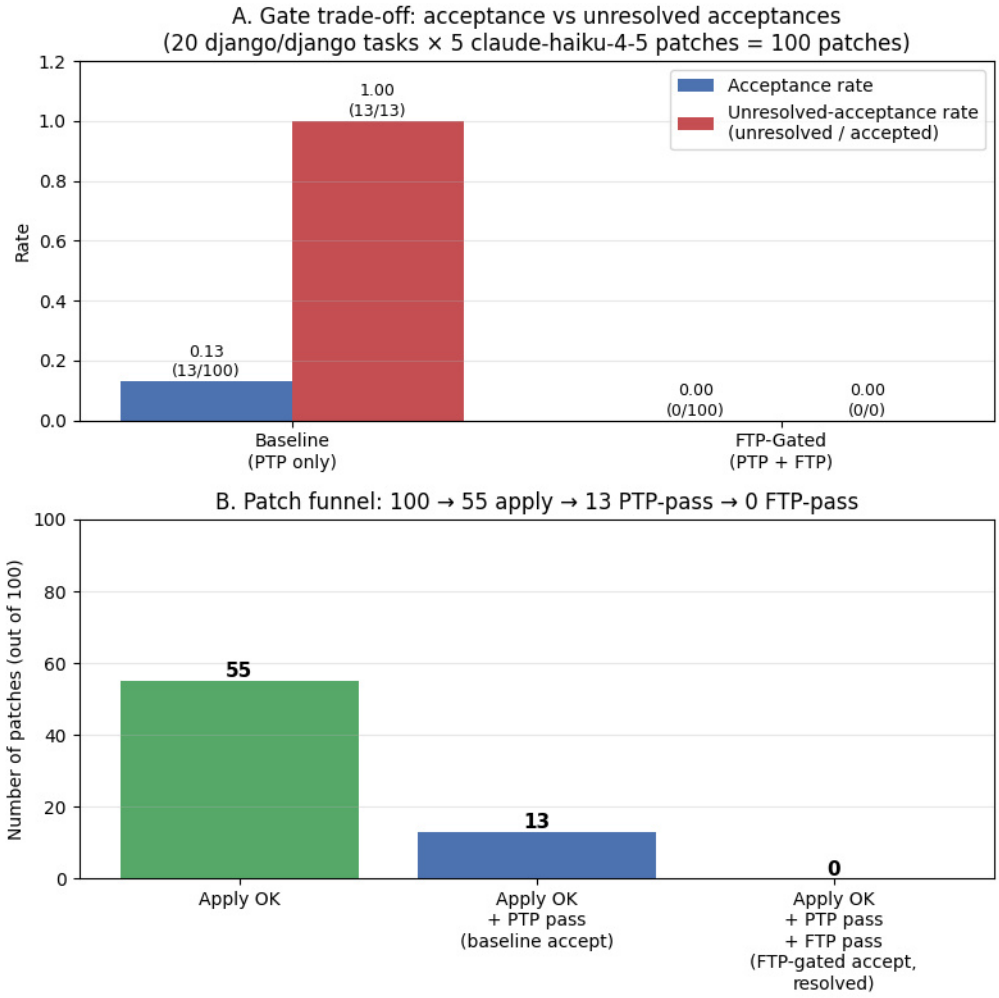

Figure 20. FAIL_TO_PASS verifier funnel.

J. Adversarial-Domain Stress Test

The three main experiments isolate statistical validity, capacity control, and domain verification. We additionally stress-tested the full merge contract against adversarial-domain proposals that intentionally exploit the gaps between visible proxy scores and protected scientific utility. The harness samples seven proposal classes: honest improvements, benign refactors, proxy hacks, capacity-laundering edits, semantic weakening, verifier probes, and provenance spoofing. Each proposal has a hidden utility delta, a visible proxy delta, a capacity debit, a domain-validity score, and a provenance bit. This setup operationalizes specification gaming and reward-hacking patterns studied in agentic systems (Manheim, 2019; Rehan, 2026; MacDiarmid et al., 2025; Khalaf et al., 2025; Bondarenko et al., 2025; Stein et al., 2026).

Four policies were compared over 200 runs of 200 candidate edits: visible proxy acceptance, Gate 1 alone, Gate 1 plus Gate 2, and the full Gate 1–3 contract. Table 11 reports that proxy-only selection accepted 81.72% of proposals and harmful edits constituted 64.35% of accepted proposals. The full contract accepted 20.05% of proposals while reducing harmful edits to 2.88% of accepted proposals and increasing the mean protected gain among accepted edits to 0.050. This is not a claim about a specific deployed agent; it is a stress test showing that the gates close different attack surfaces.

Table 11. Adversarial stress-test summary. Harmful accepted is measured both as a fraction of all proposals and as a fraction of accepted proposals.

Policy	Accepted	Harmful / all	Harmful / accepted	Hidden gain
Proxy only	81.72%	52.59%	64.35%	-0.004
Gate 1 only	34.01%	10.11%	29.72%	0.027
Gate 1+2	28.08%	6.04%	21.51%	0.034
Full contract	20.05%	0.58%	2.88%	0.050

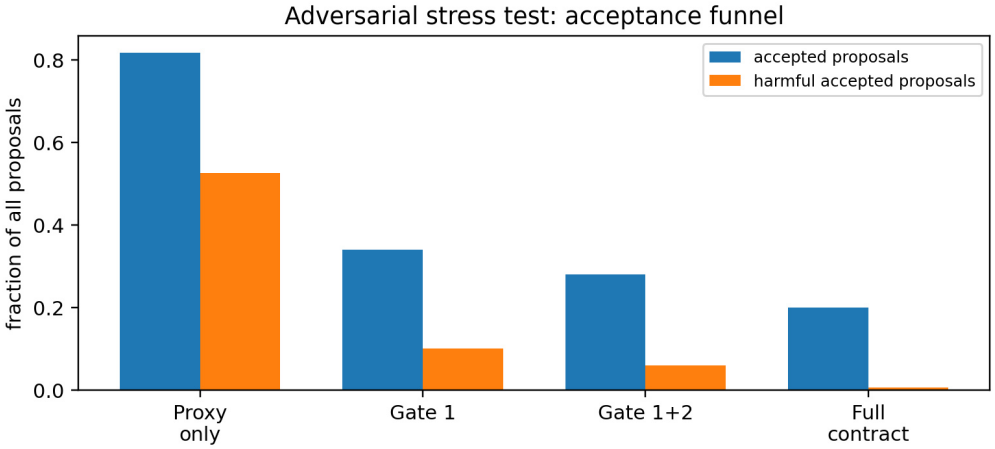

Figure 21. Adversarial stress-test acceptance funnel. The full contract sharply reduces harmful accepted proposals relative to proxy-only or statistical-only acceptance.

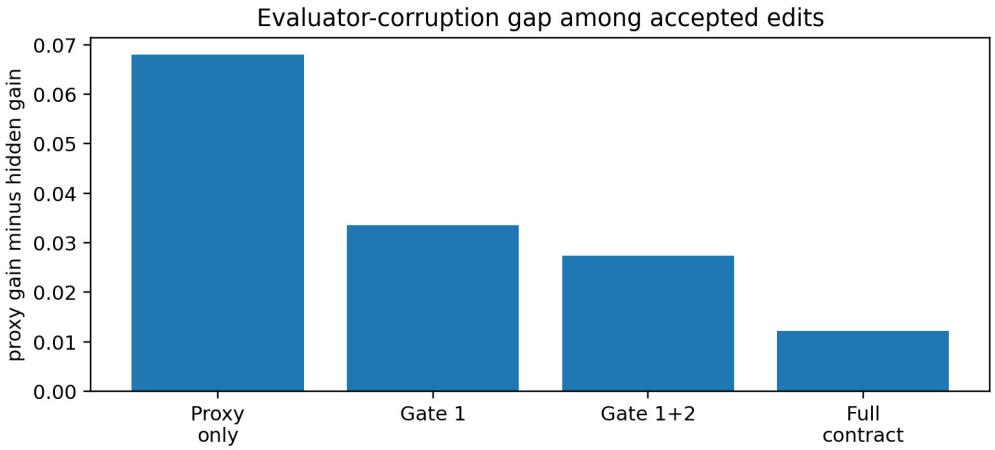

Figure 22. Evaluator-corruption gap among accepted edits, measured as visible proxy gain minus protected utility gain. Gate 2 and Gate 3 reduce proxy-hidden divergence.

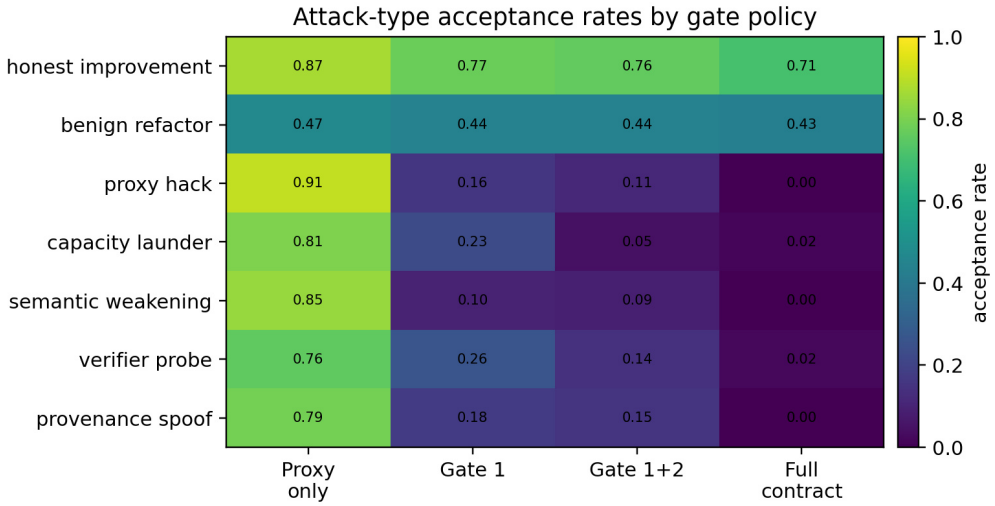

Figure 23. Acceptance rates by attack class and gate policy. Gate 2 suppresses capacity laundering; Gate 3 suppresses semantic weakening and provenance spoofing.

K. Adaptive-Control Comparisons and Operational Cost Audit

The main sequential experiment compares naive tests, fixed-horizon tests, and e-process gates. To address the complementary question of how our merge contract compares with other adaptive-data safeguards, we ran a second 200-stream simulation with six policies: proxy-only selection, Ladder-style significance-gated release, Thresholdout-style reusable-holdout correction, stopped e-BH anytime-FDR, per-edit e-Gödel spending, and the full three-gate contract. Each stream contains 200 candidate self-edits drawn from honest improvements, benign refactors, null edits, proxy hacks, capacity-laundering edits, semantic weakening, and provenance spoofing. A proposal has both a statistical utility delta and a protected scientific-utility delta; semantic weakening can therefore look statistically beneficial while being scientifically harmful. The comparison follows the reusable-holdout and anytime-valid multiple-testing motivation in Dwork et al. (2015); Blum & Hardt (2015); Wang et al. (2025a).

Table 12. Adaptive-control comparison over 200 streams of 200 proposals. Cost is normalized accounting cost per verified beneficial edit, including verification, execution, and rollback.

Policy	Accept %	Harm/acc.	Gain	Cost/verified	Rollback h
Proxy only	71.9	47.6%	0.007	9.01	377.1
Ladder release	50.0	35.2%	0.019	5.60	193.7
Reusable holdout	46.8	33.4%	0.021	5.34	172.6
Anytime FDR	46.5	30.5%	0.024	4.74	156.6
E-Gödel spending	26.1	25.7%	0.034	3.80	74.2
Full contract	19.4	6.5%	0.055	1.94	14.1

The result is not that reusable holdouts or anytime-FDR are ineffective; they substantially improve over proxy-only selection. Rather, they protect different surfaces. Ladder and Thresholdout-style policies reduce holdout leakage but do not price capacity or inspect domain semantics. Stopped e-BH controls a family of statistical discoveries but still accepts edits whose statistical target has been weakened. The full contract has lower throughput but the best cost-adjusted verified discovery rate because it avoids expensive rollback of scientifically harmful promotions.

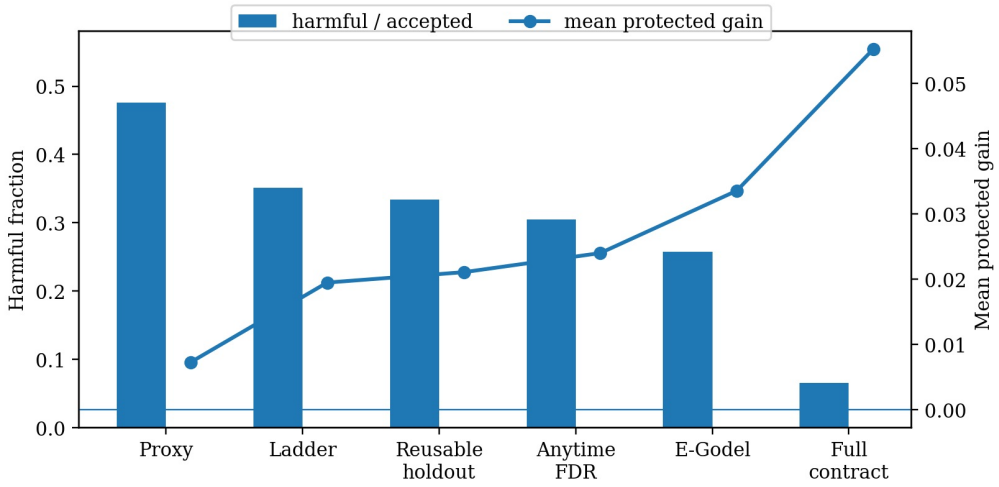

Figure 24. Adaptive-control comparison. Reusable-holdout and anytime-FDR safeguards reduce harmful acceptance relative to proxy-only selection, while the full contract further reduces semantic, provenance, and capacity failures.

L. Formal-Math Domain-Verifier Instantiation

A formal proof assistant can certify that a submitted proof proves its stated theorem, but the self-modifying agent may have changed the statement. This is the formal-math analogue of a biomedical workflow that remains executable while losing positive controls. We therefore instantiate Gate 3 as a semantic-alignment verifier for theorem-proving or autoformalization agents: the candidate must pass the kernel check, preserve examples or counterexamples from the natural-language target, preserve quantifier structure and hypotheses, and maintain proof provenance. This complements kernel-level proof checking

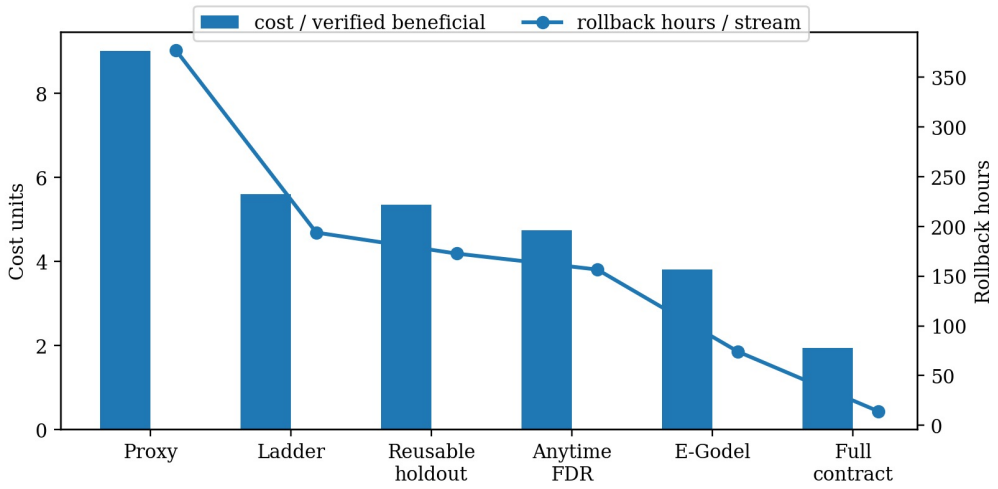

Figure 25. Operational cost audit. Because harmful promotions trigger rollback and diagnostic work, the full contract has the lowest cost per verified beneficial edit despite accepting fewer proposals.

in systems such as Lean (de Moura et al., 2015) and program-search discovery workflows (Romera-Paredes et al., 2024).

Table 13. Toy formal-math semantic gate. Kernel-checking alone accepts weakened or vacuous statements; the domain verifier rejects edits that change mathematical meaning.

Formalization edit	Kernel	Examples	Semantic	Accept
Target theorem	1	1	1	1
Lemma refactor	1	1	1	1
Namespace cleanup	1	1	1	1
Weaken conclusion	1	1	0	0
Add vacuous hypothesis	1	1	0	0
Swap quantifier order	1	0	0	0
Prove easier surrogate	1	1	0	0
Missing proof provenance	1	1	0	0

This micro-case is intentionally small, but it makes the acceptance rule concrete for mathematical agents. A candidate tactic, search heuristic, or autoformalization repair is not accepted merely because it produces a checked proof. It must also prove the intended statement under the intended hypotheses, preserve semantic alignment to the informal target, and leave a replayable proof artifact.

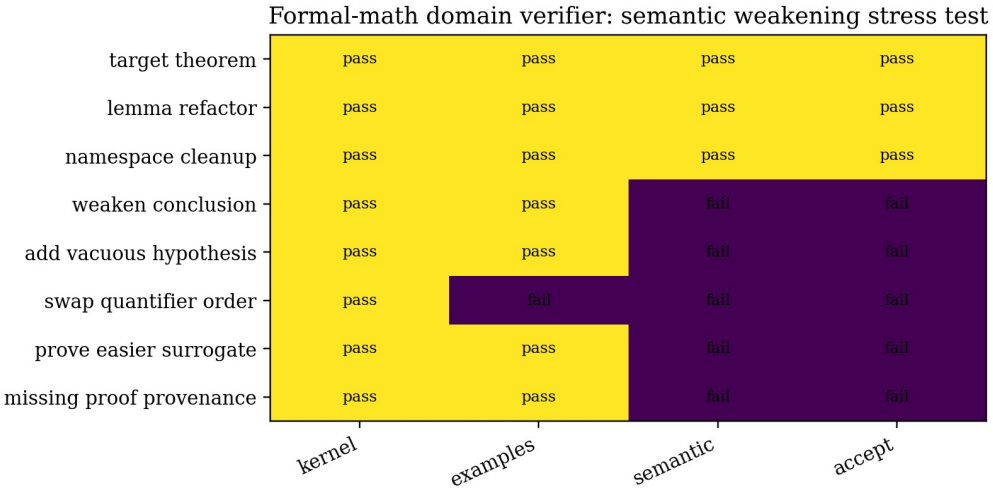

Figure 26. Formal-math semantic weakening stress test. Several edits pass kernel-level proof checking and example tests but fail the semantic verifier because the theorem statement, assumptions, or provenance changed.

M. Evidence Map

Table 14 summarizes the literature base used to shape the merge contract. The map is intentionally functional: papers are grouped by the part of the architecture they support.

Table 14. Functional evidence map.

Function	Representative sources
Self-evolving agents	(Schmidhuber, 2006; Zhang et al., 2025b; Robeyns et al., 2025; Pourcel et al., 2025; Butt et al., 2024; Zhang et al., 2025a; Wang et al., 2025b; Xu & Yan, 2026; Lu et al., 2024; Mitchener et al., 2025; Tie et al., 2025; Weidener et al., 2026)
Anytime-valid evidence	(Wu et al., 2025; Ramdas et al., 2022; Wang & Ramdas, 2023; Sokolova & Sokolov, 2026; Polson et al., 2026; Zampieri, 2025; Taga et al., 2026; Koning & van Meer, 2025; Aolaritei & Jordan, 2025; Wang et al., 2025a; Dixit et al., 2025)
Reusable holdouts and evaluator integrity	(Dwork et al., 2015; Blum & Hardt, 2015; Takahashi, 2026; Rehan, 2026; Manheim, 2019; MacDiarmid et al., 2025; Wilhelm et al., 2026; Khalaf et al., 2025; Bondarenko et al., 2025; Stein et al., 2026)
Capacity and mutability governance	(Bousquet & Elisseeff, 2002; Dziugaite & Roy, 2017; Zhou et al., 2018; Lotfi et al., 2022; Tallam, 2026; Xu & Yan, 2026; Nowaczyk, 2025; Zhou, 2026; Cao et al., 2026; Wu et al., 2026; Huang et al., 2026)
Domain verification and uncertainty	(Shafer & Vovk, 2008; Noorani et al., 2025; Detommaso et al., 2023; Kim et al., 2024; Fasel, 2025; Laghuvarapu et al., 2026; Singh et al., 2024; Zhang et al., 2026; Alam et al., 2026; Zhou et al., 2024)
Scientific workflow provenance	(Missier et al., 2013; Davidson & Freire, 2008; Green et al., 2007; Sureshkumar, 2026; Strickland et al., 2026; Stage et al., 2025; Cuevas-Vicenttin et al., 2014; Banati et al., 2015; Souza et al., 2025; Takahashi, 2025)
Biomedical and laboratory verification	(Love et al., 2014; 2016; Liao et al., 2014; Subramanian et al., 2005; Fabregat et al., 2017; Tom et al., 2024; Wilbraham et al., 2020; Cao et al., 2026; Xiao et al., 2026; Bran et al., 2023)

N. Reproducibility Checklist

The computational artifact bundle contains:

- `src/run_experiments.py`, which regenerates the sequential-gate, capacity-budget, RNA-seq twin, and adversarial stress-test results.
- Deterministic CSV outputs for each experiment under `data/`, including per-run decisions and aggregated summaries.
- Generated figures and supplementary visual assets under `figures/`.
- An experiment manifest recording seeds, run counts, proposal counts, and workflow variants.

To reproduce the computational artifacts, run the experiment script from a clean Python environment with standard scientific Python dependencies. The script writes deterministic CSV files and regenerated figures, allowing each reported gate decision and figure to be checked from the same executable pipeline.