HARNESSQUOTIENT: COUNTEREXAMPLE-GUIDED BEHAVIORAL QUOTIENTING FOR SELF-EVOLVING AGENT HARNESSES

Anonymous authors
Paper under double-blind review

ABSTRACT

Self evolving agent harnesses accumulate tools, skills, specialist agents, memory operations, and runtime policies. As the harness grows, many capabilities become redundant in appearance, but similarity is not enough to establish that two behaviors are interchangeable. Procedures can agree on common cases and still diverge in a rare context that matters for future execution. Compressing such procedures into one semantic action removes a distinction the agent still needs, a failure we call *distinction forgetting*. We introduce HARNESSQUOTIENT, a framework that treats every proposed merge as a behavioral claim to be tested. Candidate pairs are identified from inexpensive structural and semantic signals. A semantic challenger then proposes conditions under which the pair may disagree, and a planner spends a finite verification budget on valid contexts that are most informative about those possibilities. Both capabilities are executed from matched state. A discovered difference becomes a persistent separating witness, while negative evidence updates a risk-controlled certificate. The system merges only when the remaining risk is sufficiently small and otherwise leaves the pair unchanged. Semantic classes require evidence against all existing members, and certificates are reconsidered when the harness changes. This view turns harness compression into a problem of preserving the right distinctions while removing the unnecessary ones.

1 INTRODUCTION

Large language model agents increasingly depend on the harness that surrounds the model. Tools determine what the agent can execute. Memory determines what can persist across tasks. Reusable skills encode procedures. Specialist agents divide work. Verifiers judge completion. Recovery policies decide what happens after failure. As these mechanisms become central to agent performance, recent work has begun to optimize the harness itself rather than treating it as fixed infrastructure (Lee et al., 2026b; Lin et al., 2026a; Zhang et al., 2026c;b). The resulting systems can revise prompts, compose tools, retain experience, and change control logic over time.

This progress creates a problem that is easy to overlook. A harness that continually acquires capabilities also acquires distinctions. Some of those distinctions are essential. Others are accidental consequences of implementation. Two tools may expose different names while carrying out the same operation. Two skills may differ internally while presenting the same behavior to the model. Conversely, two procedures may look nearly identical in their descriptions and historical traces yet behave differently in a rare context. As the harness grows, the agent must therefore answer a difficult question before it can compress or reorganize its interface: which distinctions are real, and which can be safely removed?

The question matters because accumulation is not harmless. A larger interface increases the burden of routing and selection, and recent evidence shows that harness growth can reduce performance on tasks that were previously solved (Ke et al., 2026). Skill libraries also contain redundancy and routing failures that become more pronounced as the library expands (Ling et al., 2026; Zhang et al., 2026a). A self evolving harness therefore cannot improve indefinitely by adding more tools, skills, and policies. At some point it must consolidate what it has learned.

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.

Consolidation is dangerous for a different reason. Suppose two spreadsheet verification routines agree on ordinary workbooks, share the same signature, and pass the same regression suite. One of them, however, preserves formulas inside merged cells while the other silently replaces them with values. A similarity based deduplicator would have strong reasons to merge the two routines. A conventional regression suite might also approve the merge if the relevant workbook structure never appears. Yet the merge would remove a distinction that changes behavior. The harness would become simpler, but less faithful to the capabilities it previously possessed.

We call this failure *distinction forgetting*. It does not require deleting memory or changing model weights. The loss occurs when the harness assigns one semantic identity to behaviors that should remain distinguishable. This observation changes how compression should be approached. Similarity can identify plausible merge candidates, but it cannot justify a merge. The decisive evidence is behavioral. Before collapsing two capabilities, the system should actively search for a valid context in which they disagree.

This leads to a different view of harness evolution. A proposed merge is not a bookkeeping operation. It is a claim of behavioral interchangeability, and such a claim should be challenged before it is accepted. Historical agreement provides evidence in favor of the claim. A separating case refutes it. Failure to find such a case raises confidence, but should not be mistaken for proof. The harness should retain the option to abstain when the available evidence is insufficient.

We introduce HARNESSQUOTIENT, a framework that learns a compact semantic interface by testing these interchangeability claims. The process begins with inexpensive signals such as signatures, descriptions, dependencies, call patterns, and execution traces to identify candidate pairs. For each pair, a semantic challenger proposes conditions under which the two behaviors might differ. The system then spends a finite verification budget on valid contexts chosen to cover the most plausible sources of disagreement. Both capabilities are executed from matched state, and their observable effects are compared. When a difference appears, the context is stored as a persistent *separating witness*. When repeated tests fail to reveal a difference, the accumulated evidence supports a calibrated merge decision. If confidence remains too low, the pair stays separate.

The same principle governs the construction of larger semantic classes. Pairwise approval alone is not enough because approximate behavioral agreement need not propagate safely through a chain of capabilities. HARNESSQUOTIENT therefore requires a new member to be supported against the class it is joining, while previously discovered separating witnesses remain permanent constraints unless later changes invalidate them. Certificates are tied to the behavior versions and environments that produced them, so subsequent harness changes trigger revalidation instead of inheriting old conclusions without examination.

The resulting perspective is simple. Harness growth should increase useful capability without forcing the semantic interface to grow at the same rate. Achieving that goal requires more than compression. It requires a disciplined account of which distinctions may be forgotten and which must remain available to the agent.

Our contributions are as follows:

- We identify *distinction forgetting*, a failure mode in which harness compression removes a behavioral difference that remains necessary for correct execution.
- We formulate harness compression as the recovery of behavioral equivalence classes among existing capabilities, with separation, merging, and abstention determined by evidence rather than similarity alone.
- We introduce counterexample guided verification that turns plausible sources of disagreement into valid test contexts and allocates a finite test budget according to their posterior value.
- We preserve discovered separating witnesses and require class wide certification, allowing the semantic interface to contract while guarding against unsafe chaining and stale equivalence claims.

In this view, abstraction is a disciplined form of forgetting. A self evolving harness improves not only by learning what to add, but by learning which distinctions it has earned the right to remove.

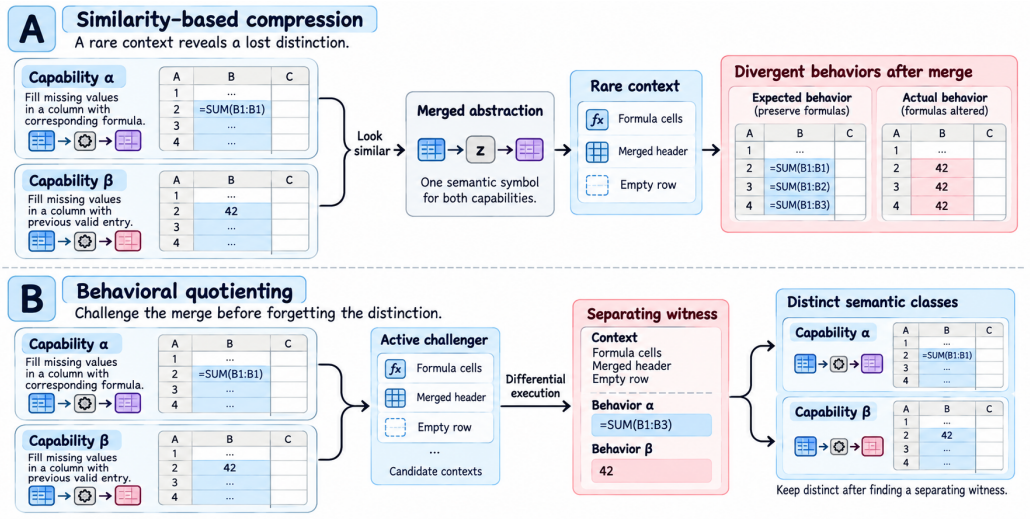


Figure 1: Motivation for behavioral quotienting. Panel A shows how similarity based compression can collapse two capabilities that agree on common cases yet diverge in a rare context, which removes a distinction the harness still needs. Panel B shows the alternative adopted by HARNESSQUOTIENT: before merging, the system actively searches for informative valid contexts and preserves any discovered separating witness as evidence that the two capabilities should remain distinct.

2 RELATED WORK

2.1 SELF-EVOLVING AGENT HARNESSES

Recent work increasingly treats the machinery surrounding an agent as part of the system to be optimized. Meta-Harness searches over executable harness code using traces, scores, and previous candidates (Lee et al., 2026b). Agentic Harness Engineering and Self-Harness revise persistent harness components from execution evidence (Lin et al., 2026a; Zhang et al., 2026c). Recursive Harness Self-Improvement studies repeated refinement of harness instructions (Lee et al., 2026a), while Natural-Language Agent Harnesses represents control logic as portable textual artifacts (Pan et al., 2026). JIT-Agent synthesizes harness modules conditioned on the task (Zhang et al., 2026b). HarnessDev evaluates whether systems can create and revise runnable harness infrastructure (Wu et al., 2026), and related work separates the quality of a harness update from the solver's ability to exploit that update (Lin et al., 2026b).

As harnesses become persistent and mutable, evaluation must also account for what an update preserves. HarnessLens directs verification toward behaviorally relevant tasks rather than replaying an undifferentiated suite (Xu et al., 2026). EVOHARNESSBENCH places change inside the harness itself and measures retention as tools, skills, and agents evolve (Ke et al., 2026). These works establish that harness evolution is a continual process rather than a sequence of isolated improvements. Our focus is complementary. We study the decision that arises when an evolving harness tries to simplify itself by assigning one semantic identity to capabilities that appear redundant. The difficulty is that historical success and structural similarity may fail to reveal a context in which the capabilities behave differently. This turns safe compression into a problem of discovering which distinctions must be preserved.

2.2 BEHAVIORAL ABSTRACTION AND COUNTEREXAMPLE GUIDED VERIFICATION

Reusable skill systems place complex procedures behind compact interfaces, but growing libraries can accumulate redundancy, routing ambiguity, and overlapping functionality (Ling et al., 2026; Zhang et al., 2026a). SkillZip is especially close to our setting. It compresses execution graphs while preserving declared boundary signatures, dependencies, verifier reachability, and reversible expansion, and it revises risky macros as the library changes (Tan et al., 2026). These safeguards preserve known procedural structure. They do not establish that two implementations remain

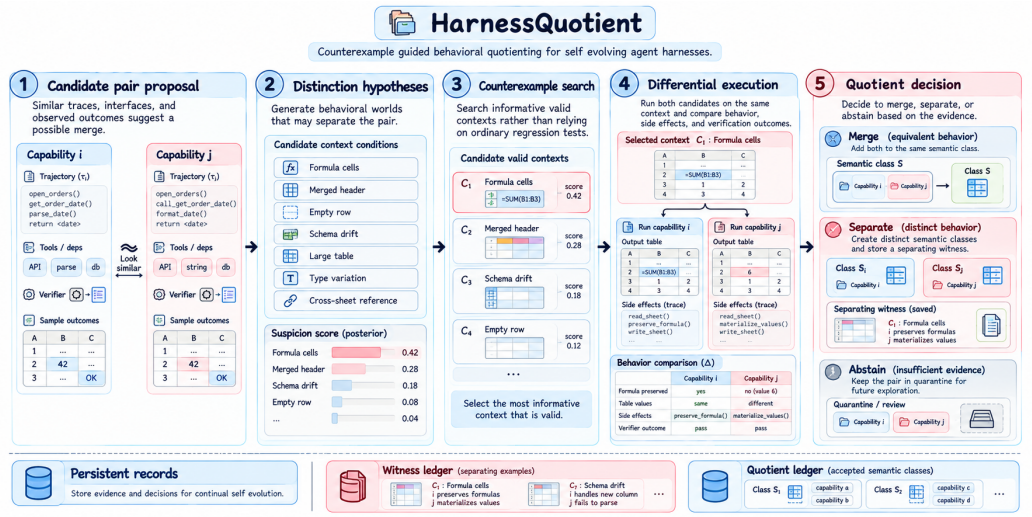


Figure 2: Overview of HARNESSQUOTIENT. The system first proposes a candidate pair from inexpensive structural evidence, then constructs distinction hypotheses that describe plausible ways the pair may differ. A planner allocates a finite verification budget to valid contexts that are expected to be most informative. Both candidates are executed from matched state, their observable effects are compared, and the result is used to merge, separate, or abstain. Certificates and separating witnesses are retained as persistent records so that later harness changes can trigger targeted revalidation.

behaviorally interchangeable on contexts that the declared structure does not describe. We therefore treat structural compatibility as evidence for proposing a merge, not as proof that the merge is sound.

This view connects harness compression to behavioral abstraction. Classical state aggregation and bisimulation preserve distinctions only when they matter to future behavior (Givan et al., 2003). We apply the same principle to harness capabilities. Two capabilities may share a semantic identity only when the behaviors relevant to the agent cannot distinguish them. The central challenge is therefore not representation alone, but evidence. Counterexample guided synthesis tests a candidate by searching for evidence that invalidates it (Solar-Lezama et al., 2006), while property based testing generates executions that expose violations missed by ordinary examples (Claessen & Hughes, 2000). Recent agent systems use counterexamples to refine language conditioned artifacts, including automata in CEDAR and regular expressions in A-CEGIS (Chen et al., 2026; Badrinarayan & Parthasarathy, 2026). HARNESSQUOTIENT uses the same spirit of falsification for a different object: a proposed equivalence between existing capabilities. A separating witness is not merely feedback for repairing one candidate. It is durable evidence that a semantic distinction must remain available to the harness. When no witness is found, the accumulated negative evidence can support a calibrated merge decision, while unresolved pairs remain separate.

3 METHOD

3.1 PROBLEM FORMULATION

At evolution stage t, let the harness contain a set of callable behaviors

$$\mathcal{Z}_t = \{z_1, \ldots, z_n\}. \tag{1}$$

A behavior may be a tool, a reusable skill, a specialist agent, a memory operation, or a compiled procedure. The physical set can grow over time. Our objective is to expose a smaller semantic interface to the model without identifying two behaviors whose difference can affect execution.

Behavior is always judged in context. Let $x \in \mathcal{X}_t$ denote a valid execution context and let

$$\Omega(z, x) = (y, \Delta s, e, v) \tag{2}$$

be the observable signature of behavior z in context x. The signature contains its return value y, the externally relevant change in state Δs, visible side effects e, and a verifier or bounded downstream

outcome v. Runtime cost can be tracked separately when needed for dispatch. A comparator Δ measures whether two signatures differ in a way that matters to the application.

A context separates two behaviors when

$$\Delta(\Omega(z_i, x), \Omega(z_j, x)) > \tau. \tag{3}$$

Such a context is a *separating witness*. Behavioral equivalence in this paper is therefore relative to the valid context set $\mathcal{X}_t$, the observation function Ω, and the tolerance τ. Claims are confined to this scope.

The semantic interface is represented by a partition Π_t of $\mathcal{Z}_t$. Each block corresponds to one action presented to the model, while its physical implementations remain available to the runtime. Let $R_t(i, j)$ denote the posterior risk that z_i and z_j are not behaviorally equivalent, and let $\mathcal{W}_{ij}^{-}$ denote their stored separating witnesses. We seek a compact partition subject to the condition that every pair placed in the same block has sufficiently low residual risk and no surviving separating witness:

$$\min_{\Pi_t} \quad |\Pi_t| \tag{4}$$

$$\text{s.t.} \quad R_t(i, j) \le \delta, \qquad \forall C \in \Pi_t,\ z_i, z_j \in C,\ i \neq j, \tag{5}$$

$$\mathcal{W}_{ij}^{-} = \varnothing, \qquad \forall C \in \Pi_t,\ z_i, z_j \in C,\ i \neq j. \tag{6}$$

This formulation makes compression subordinate to behavioral preservation. A demonstrated difference blocks a merge. Uncertainty may also block a merge even when no difference has yet been observed.

3.2 Candidate Pairs and Distinction Hypotheses

Testing every pair is wasteful. We first identify plausible candidates using inexpensive evidence such as signatures, descriptions, dependencies, call sites, trajectory overlap, and verifier structure. These signals determine where verification effort should be spent. They do not authorize merging.

For a candidate pair (z_i, z_j), the next task is to identify plausible ways in which the pair could disagree. A semantic challenger receives the available documentation, implementation summaries, traces, dependency information, and historical failures. It returns a collection

$$\mathcal{G}_{ij} = \{g_1, \ldots, g_m\}, \tag{7}$$

where each g describes a family of contexts in which a behavioral difference may become visible. Examples include non idempotent operations, ignored generated files in a repository, formulas inside merged spreadsheet cells, or a memory write whose effect appears only on later retrieval.

These distinction conditions define a finite family of behavioral worlds

$$\Theta_{ij} = \{\theta_0, \theta_1, \ldots, \theta_M\}. \tag{8}$$

The world θ_0 represents behavioral equivalence within the scope of the study. Every other world describes one possible pattern of disagreement across valid contexts. A world may include several distinction conditions, since multiple failure modes can coexist. Semantic evidence s induces an initial posterior

$$p_0(\theta) = P(\theta \mid s). \tag{9}$$

The same evidence therefore affects both the distribution of plausible disagreements and the remaining probability assigned to equivalence.

Every proposed test context must satisfy a validity predicate $V(x) = 1$. This restriction matters because a challenger that is allowed to invent impossible states can manufacture disagreements with no operational meaning.

3.3 Choosing Counterexample Tests

The verification budget should be spent on contexts that are most likely to resolve uncertainty. After k tests, let $p_k(\theta)$ denote the current posterior. For a non equivalence world θ, define

$$q_\theta(x) = P(\Delta(\Omega(z_i, x), \Omega(z_j, x)) > \tau \mid \theta, x)\,. \tag{10}$$

This quantity describes the chance that valid context x exposes a material difference if world θ is true.

For a batch S of valid tests, we score the posterior mass expected to be exposed by at least one member of the batch:

$$F(S) = \sum_{\theta \neq \theta_0} p_k(\theta) \left[1 - \prod_{x \in S} (1 - q_\theta(x)) \right]. \tag{11}$$

Given a budget B, the planner chooses

$$\max_{S \subseteq \mathcal{X}_t} F(S) \qquad \text{s.t.} \quad |S| \leq B,\ V(x) = 1\ \forall x \in S. \tag{12}$$

The objective is a weighted probabilistic coverage function. It is nonnegative, monotone, and submodular, so greedy marginal gain selection obtains the standard $(1 - 1/e)$ approximation under a cardinality budget (Nemhauser et al., 1978). When every $q_\theta(x)$ is binary, the objective reduces to posterior weighted maximum coverage.

This separation of roles is useful. The semantic challenger proposes meaningful hypotheses and valid contexts. The planner decides how to distribute a limited verification budget. Neither component is given the hidden truth about whether the pair is equivalent.

Tests may also be selected sequentially. After a negative result, the posterior changes and the next context is chosen using the updated marginal gains. A negative observation reduces the probability of worlds it was capable of exposing, but does not remove them unless the observer is known to have no false negatives.

3.4 Matched Differential Execution

Each selected context is instantiated twice from matched state. Whenever the environment permits, the executions share the same workspace snapshot, random seed, environment version, fixtures, and relevant memory state. The two behaviors are then run independently and compared through Ω.

For test k, define

$$y_k = \mathbf{1}[\Delta(\Omega(z_i, x_k), \Omega(z_j, x_k)) > \tau]. \tag{13}$$

If $y_k = 1$, the pair is separated and the record

$$w_k = (x_k, \Omega(z_i, x_k), \Omega(z_j, x_k)) \tag{14}$$

becomes part of the persistent witness set $\mathcal{W}_{ij}^{-}$. The witness records a concrete reason why the two semantic identities must remain distinct.

The observation function may include a bounded downstream probe when immediate outputs are insufficient. Two editing operations can return the same success message while leaving different repository states. Two memory operations can acknowledge the same write while changing future retrieval differently. The comparator should therefore reflect the consequences that matter to later agent behavior rather than surface text alone.

3.5 Certification and Abstention

Not finding a witness is informative, but it is not a proof of equivalence. After every observation, we update the posterior over behavioral worlds:

$$p_{k+1}(\theta) \propto p_k(\theta)\, P(y_k \mid \theta, x_k). \tag{15}$$

The likelihood model includes the measured false positive and false negative rates of the differential observer. These rates can be estimated from known equivalent implementation pairs, deliberately perturbed non equivalent pairs, or application specific fixtures.

After at most B tests, the remaining risk is

$$R_t(i, j) = 1 - P(\theta_0 \mid D_B), \tag{16}$$

where $D_B = \{(x_k, y_k)\}_{k=1}^{B}$ is the accumulated evidence. The pairwise decision is

$$\textsc{Decision}(i, j) = \begin{cases} \mathsf{Sep}, & \mathcal{W}_{ij}^{-} \neq \varnothing, \\ \mathsf{Eq}, & R_t(i, j) \leq \delta, \\ \mathsf{Abstain}, & \text{otherwise.} \end{cases} \tag{17}$$

An abstained pair remains separate and may receive more verification budget later. This third outcome is important because it distinguishes lack of evidence for a difference from sufficient evidence for a merge.

The numerical certificate is meaningful only with respect to the chosen behavioral worlds, context generator, observation function, and calibrated likelihoods. We therefore treat the certificate as a scoped statistical claim rather than a universal proof of semantic equivalence.

3.6 Building the Quotient

Pairwise certification does not justify unrestricted transitive closure. If z_1 is close enough to z_2, and z_2 is close enough to z_3, it does not follow that z_1 and z_3 should share a semantic identity. An error in one pairwise decision could otherwise collapse an entire connected component.

We use complete linkage admission. Let C be an existing semantic class. A new behavior z may join C only if

$$\textsc{Decision}(z, z') = \mathsf{Eq} \quad \text{and} \quad \mathcal{W}^{-}_{zz'} = \varnothing \qquad \forall z' \in C. \tag{18}$$

A cheap comparison with a representative can reject an obvious nonmember early, but admission requires evidence against every member of the class. This prevents a chain of plausible pairwise decisions from creating an uncertified equivalence between distant members.

Each class stores a canonical semantic symbol, its certified physical implementations, the relevant certificates, version metadata, and witnesses that define its boundary. The model interacts with the canonical symbol. The runtime chooses among compatible physical implementations according to availability, cost, and platform constraints. The physical capabilities are retained. What is reduced is the number of semantic choices presented to the model.

3.7 Maintaining the Quotient as the Harness Changes

A certificate concerns particular behavior versions in a particular execution environment. It should not survive unrelated changes by default. When an implementation, dependency, verifier, or environment contract changes, affected certificates become stale. Revalidation begins with previously informative boundary cases and stored separating witnesses, then spends new counterexample search budget where uncertainty remains.

A witness that still reproduces continues to block the corresponding merge. A new witness can split an existing class. Behaviors that were previously separated may be reconsidered after an implementation change, but only after new evidence has been collected.

The resulting policy is intentionally asymmetric. Separation can be accepted as soon as a valid witness appears. Merging requires accumulated evidence. Similarity determines where to look, counterexamples try to falsify interchangeability, calibrated evidence permits compression, and later harness changes reopen conclusions whose supporting conditions no longer hold.

4 Experiments

The experiments follow the logic of the method rather than the chronology of its components. We first ask whether appearance is sufficient to justify compression. We then ask whether semantic guidance reduces the number of executions needed to obtain behavioral evidence. Finally, we ask whether the resulting quotient remains useful when the capability library grows and whether its conclusions survive later semantic drift. The main study uses a controlled executable benchmark in which behavioral equivalence is known by construction. The public benchmark protocol, information budgets, and external baselines are specified in Appendix B.

4.1 Experimental Protocol

The controlled benchmark contains twelve binary context factors and seven distinction families. A non equivalent pair differs only under a sparse conjunction of two to four factors, while natural contexts activate each factor with probability 0.08. The semantic challenger receives noisy evidence about the pair but never the hidden distinction itself. We use 500 validation pairs and 1,200 held out

test pairs per seed over five seeds. Passive baselines use descriptions, traces, structural contracts, or a learned equivalence classifier. Active baselines use IID regression, random valid fuzzing, generic coverage, or semantic random search. Semantic random and HARNESSQUOTIENT receive the same semantic hypotheses, observer, certifier, and execution budget. Their only difference is which context is queried next. Thresholds are selected on validation data for a 2% unsafe merge target. Full generation, observer, candidate, and held out family protocols appear in Appendix A.3 and Appendix B.

4.2 A MERGE NEEDS EVIDENCE, NOT RESEMBLANCE

Table 1 exposes a basic asymmetry that an aggregate accuracy score would conceal. Description similarity, trace similarity, and the learned passive classifier appear safe because they almost never make a strong compression claim. Their unsafe merge rates stay near one percent, yet each certifies only 6–11% of genuinely equivalent pairs. In other words, these policies purchase safety by preserving nearly the entire interface. The result is a system that rarely makes a semantic mistake because it rarely commits to an abstraction.

Table 1: Passive merge policies. No candidate behavior is executed.

Method	Safe cov. ↑	Unsafe ↓	Merge
Description sim.	6.5 ± 1.1	0.8 ± 0.5	3.7 ± 0.7
Trace sim.	10.6 ± 3.3	1.1 ± 0.5	5.8 ± 1.8
Passive clf.	11.0 ± 4.2	0.4 ± 0.3	5.7 ± 2.3
Structural	99.0 ± 0.3	83.2 ± 0.5	91.1 ± 0.3
Oracle	100.0	0.0	50.1 ± 0.4

Structural matching reaches the opposite corner. It certifies 99.0% of equivalent pairs and therefore looks almost ideal if one reads only the compression column. But 83.2% of non equivalent pairs are also collapsed. The failure is revealing: a declared schema or dependency graph can say that two capabilities *look compatible*, but it cannot establish that their behavior remains interchangeable in contexts the contract never names. Structure is consequently valuable as a proposal mechanism. It is not a certificate of equivalence.

The tension matters because both easy solutions are unsatisfactory. A system can be safe by refusing to merge, or compact by deleting distinctions indiscriminately. Neither behavior is the object of this paper. A useful quotient must remove enough redundancy to simplify the interface while retaining precisely those distinctions that can alter future execution. We therefore keep safe merge coverage and unsafe merge rate separate throughout the evaluation and treat abstention as an informative outcome rather than folding it into a single classification score.

Table 2: Active verification under the same budget $B = 16$, valid context space, observer, and merge threshold. Values are mean $\pm$ 95% CI over five seeds.

Method	Witness ↑	Safe cov. ↑	Unsafe ↓	Abstain ↓	Probes ↓
IID regression	7.0 ± 1.1	0.0	0.00	95.8 ± 0.6	15.7 ± 0.1
Random fuzzing	52.1 ± 2.0	0.0	0.00	72.9 ± 1.2	13.5 ± 0.1
Generic coverage	66.2 ± 1.4	0.0	0.00	65.9 ± 1.2	12.8 ± 0.1
Semantic random	83.0 ± 2.0	21.3 ± 1.5	0.36 ± 0.23	46.9 ± 0.8	10.2 ± 0.1
HarnessQuotient	89.2 ± 1.1	35.3 ± 1.3	0.60 ± 0.48	36.5 ± 0.4	8.7 ± 0.1

Once execution evidence is permitted, the question becomes sharper: how much behavioral information can a method obtain before the verification budget is exhausted? Table 2 compares all active policies under the same sixteen-execution budget, valid context space, observer, and merge threshold. IID regression recovers only 7.0% of hidden distinctions. Random fuzzing reaches 52.1%, and generic coverage reaches 66.2%. The jump from generic coverage to semantic random, from 66.2% to 83.0%, isolates the value of pair-specific hypotheses. The remaining gap to HARNESSQUOTIENT, which reaches 89.2%, isolates the value of choosing the next context deliberately rather than sampling from those hypotheses without a plan.

That final gap is small enough to be believable and large enough to matter. Relative to semantic random, active selection adds 6.2 percentage points of witness recall with a 95% seed-level interval of $[4.0, 8.5]$, adds 14.0 points of safe merge coverage with interval $[13.1, 14.9]$, and saves 1.51 executions per pair. The unsafe merge difference is only 0.24 points and its interval crosses zero.

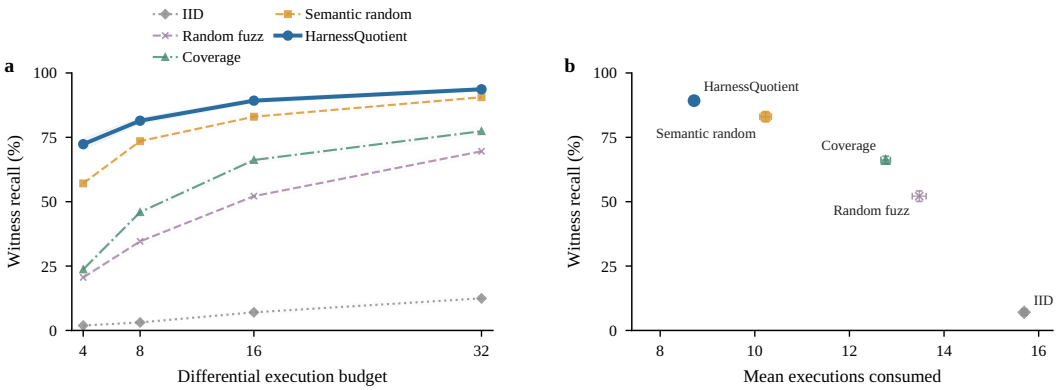


Figure 3: Verification efficiency under matched information and execution budgets. **a**, witness recall as the budget grows. **b**, witness recall against the number of executions actually consumed at budget 16. Error bands and bars denote 95% confidence intervals across seeds.

The planner is therefore not buying coverage by weakening the safety rule. It is reaching decisive evidence sooner. Semantic hypotheses identify where disagreement may live; posterior coverage determines which uncertainty should be resolved first.

4.3 Active Search Matters Most When Verification Is Scarce

Figure 3 shows where planning earns its keep. With only four executions, HarnessQuotient finds 72.3% of hidden distinctions, compared with 57.1% for semantic random and 23.7% for generic coverage. The gap narrows as the budget grows, which is the signature of a search advantage rather than an oracle advantage. The efficiency view tells the same story in systems terms: semantic hypotheses move the operating point upward, and active posterior coverage moves it further upward and leftward. The benefit is therefore not more permissive certification, but better ordering of expensive tests before exhaustive exploration becomes affordable.

4.4 Compression Must Survive Growth and Change

The quotient remains useful only if compactness does not conceal semantic damage. At 1,000 capabilities, HarnessQuotient reduces the exposed interface by 14.8% while keeping unsafe behavior failure at 0.03%; structural compression removes more of the interface, but incurs roughly 52 times more unsafe failures. Under two later drift events, revalidation keeps stale merges at 1.03%, compared with 6.41% for a static quotient, while the witness ledger cuts repeated verification by 19.1%. The full scaling and continual analyses appear in Appendix A.5. Together, these results suggest that the useful unit of reuse is not merely a compressed action, but a compressed action together with the evidence that defines its semantic boundary.

5 Conclusion

A self-evolving harness must decide not only which capabilities to acquire, but which distinctions it can safely remove from the agent's interface. HarnessQuotient treats a proposed merge as a falsifiable behavioral claim: semantic hypotheses suggest where two capabilities may diverge, active testing spends a limited budget on informative contexts, and calibrated evidence determines whether compression is justified.

The experiments support this view while marking its boundary. Behavioral evidence yields safer, more sample-efficient compression than resemblance alone, but every certificate remains scoped to the contexts that can be generated and the effects that can be observed. The broader lesson is that abstraction should be earned by evidence, not inferred from similarity. A growing harness should learn which distinctions it can afford to forget.

REFERENCES

Sidhesh Badrinarayan and Adithya Parthasarathy. Counterexamples as feedback for agent self-correction, 2026.

Lekai Chen, Alvaro Velasquez, and Ashutosh Trivedi. Cedar: Automata as verifiable interfaces for language-guided embodied action, 2026.

Koen Claessen and John Hughes. Quickcheck: A lightweight tool for random testing of haskell programs. In *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming (ICFP)*, pp. 268–279, 2000.

Shengda Fan, Xin Cong, Yuepeng Fu, Zhong Zhang, Shuyan Zhang, Yuanwei Liu, Yesai Wu, Yankai Lin, Zhiyuan Liu, and Maosong Sun. Workflowllm: Enhancing workflow orchestration capability of large language models. In *International Conference on Learning Representations*, 2025a.

Shengda Fan, Xin Cong, Zhong Zhang, Yuepeng Fu, Yesai Wu, Hao Wang, Xinyu Zhang, Enrui Hu, and Yankai Lin. Generalizing experience for language agents with hierarchical metaflows. In *Advances in Neural Information Processing Systems*, 2025b.

Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 147(1–2):163–223, 2003.

Zixuan Ke, Vaidehi Patil, Haizhou Shi, Yang Li, Ye Liu, Sarath Shekkizhar, Anurag Koul, Jiayu Wang, Xuan Phi Nguyen, Semih Yavuz, Mohit Bansal, and Shafiq Joty. Evoharnessbench: Can your agents keep pace with an evolving harness?, 2026.

Daehee Lee, Minjong Yoo, Woo Kyung Kim, Wonje Choi, and Honguk Woo. Incremental learning of retrievable skills for efficient continual task adaptation. In *Advances in Neural Information Processing Systems*, volume 37, pp. 17286–17312, 2024.

Daehee Lee, Dongsu Lee, TaeYoon Kwack, Wonje Choi, and Honguk Woo. Policy compatible skill incremental learning via lazy learning interface. In *Advances in Neural Information Processing Systems*, 2025.

Hyunin Lee, Jinglue Xu, Jeffrey Seely, Donghyun Lee, Matei Zaharia, and Yujin Tang. Recursive harness self-improvement, 2026a.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses, 2026b.

Junlong Li, Wenshuo Zhao, Jian Zhao, Weihao Zeng, Haoze Wu, Xiaochen Wang, Rui Ge, Yuxuan Cao, Yuzhen Huang, Wei Liu, Junteng Liu, Zhaochen Su, Yiyang Guo, Fan Zhou, Lueyang Zhang, Juan Michelini, Xingyao Wang, Xiang Yue, Shuyan Zhou, Graham Neubig, and Junxian He. The tool decathlon: Benchmarking language agents for diverse, realistic, and long-horizon task execution. In *International Conference on Learning Representations*, 2026.

Jiahang Lin, Shichun Liu, Chengjun Pan, Lizhi Lin, Shihan Dou, Zhiheng Xi, Xuanjing Huang, Hang Yan, Zhenhua Han, Tao Gui, and Yu-Gang Jiang. Agentic harness engineering: Observability-driven automatic evolution of coding-agent harnesses, 2026a.

Minhua Lin, Juncheng Wu, Zijun Wang, Zhan Shi, Yisi Sang, Bing He, Zewen Liu, Tianxin Wei, Zongyu Wu, Zhiwei Zhang, Dakuo Wang, Xiang Zhang, Benoit Dumoulin, Cihang Xie, Yuyin Zhou, Suhang Wang, and Hanqing Lu. Harness updating is not harness benefit: Disentangling evolution capabilities in self-evolving llm agents, 2026b.

George Ling, Shanshan Zhong, and Richard Huang. Agent skills: A data-driven analysis of claude skills for extending large language model functionality, 2026.

George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14:265–294, 1978.

Linyue Pan, Lexiao Zou, Shuo Guo, Jingchen Ni, and Hai-Tao Zheng. Natural-language agent harnesses, 2026.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 48371–48392. PMLR, 2025.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations*, 2024.

Armando Solar-Lezama, Liviu Tancau, Rastislav Bodik, Sanjit Seshia, and Vijay Saraswat. Combinatorial sketching for finite programs. In *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2006.

Xingyu Tan, Xiaoyang Wang, Qing Liu, Xiwei Xu, Xin Yuan, Liming Zhu, and Wenjie Zhang. Skillzip: Contract-preserving graph compression for scalable agent skill libraries, 2026.

Yuhao Wu, Jingyuan Zhang, Jiajun Shi, Xinping Lei, Qingshui Gu, Yuxuan Zhang, Zexuan Wang, Chen He, Chen Huang, Maojia Song, Zhiyuan Zeng, Shaowen Wang, Jinkai Liu, Yunfeng Shi, Jiaheng Liu, Shen Yan, Wenhao Huang, Ge Zhang, and Wenxuan Zhang. Harnessdev: Can llms create and evolve their own agent harness?, 2026.

Jinghan Xu, Yikai Zhang, Aili Chen, Weiyuan Li, Jiaqing Liang, and Deqing Yang. Verify smarter, evolve further: Efficient harness evolution through behavior-aware verification, 2026.

Chi Zhang, Yimin Liu, Xinze Chen, and Ping Ji. What keeps agent skills from being reusable? evidence from 138k skill.md files, 2026a.

Guibin Zhang, Leo Lu, Fangzhou Xie, Kang Zhu, Junhao Wang, Zhifei Xie, Zhaochen Yu, Zihang Liu, Zhongxiang Sun, Qiankun Li, Yue Liao, Heng Chang, Xiaobin Hu, Qibing Ren, Wangchunshu Zhou, and Shuicheng Yan. Jit-agent: Scaling harness intelligence via just-in-time harness evolution, 2026b.

Hangfan Zhang, Shao Zhang, Kangcong Li, Chen Zhang, Yang Chen, Yiqun Zhang, Lei Bai, and Shuyue Hu. Self-harness: Harnesses that improve themselves, 2026c.

Longtao Zheng, Zhiyuan Huang, Zhenghai Xue, Xinrun Wang, Bo An, and Shuicheng Yan. Agentstudio: A toolkit for building general virtual agents. In *International Conference on Learning Representations*, 2025.

A Additional Controlled Experiments

A.1 The Safety Compression Landscape

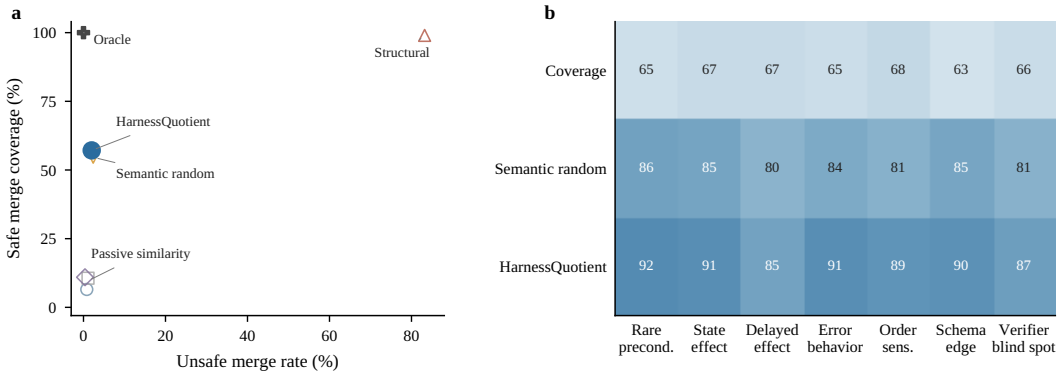


Figure 4: Where behavioral evidence changes the merge decision. **a**, safe merge coverage versus unsafe merge rate across passive and semantic policies. **b**, witness recall by distinction family at budget 16.

Figure 4a makes the central tradeoff visible without compressing it into one score. Description and trace similarity remain near the origin because they rarely act. Structural matching occupies the opposite corner: it removes almost every redundant distinction, but also many distinctions that later change behavior. Semantic random and HarnessQuotient inhabit the smaller region in which useful compression coexists with low semantic risk. This is the regime the method is designed to enlarge.

The family breakdown in Figure 4b shows that the result is not carried by one convenient mutation. HarnessQuotient exceeds 90% recall on rare preconditions, state effects, error behavior, and schema edges, while remaining above 85% on delayed effects and verifier blind spots. The last two are deliberately harder because the evidence that separates the pair is delayed or partially hidden from the immediate verifier. Generic coverage is comparatively flat because its tests are independent of the candidate pair.

A.2 Robustness to Evidence Quality and Observer Noise

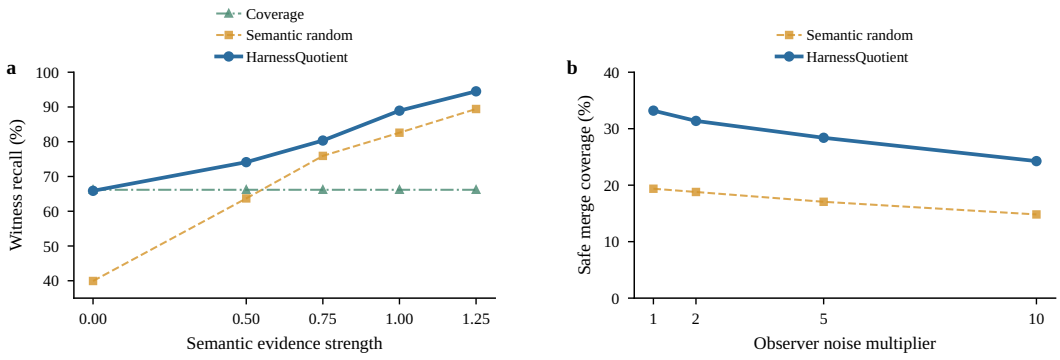


Figure 5: Robustness at budget 16. **a**, witness recall as semantic evidence is weakened or strengthened. **b**, safe merge coverage as the differential observer becomes noisier.

The semantic challenger should help only when its clues carry information. Figure 5a shows exactly this limiting behavior. When the semantic signal is removed, HarnessQuotient falls to 65.9%

witness recall, essentially the same as generic coverage at 66.2%. As the signal strengthens, recall rises smoothly to 94.5%. The planner therefore does not retain a hidden advantage after its pair specific information is erased.

Observer noise produces a different response. At ten times the default false positive and false negative rates, witness recall remains 88.0%, while safe merge coverage falls from 33.2% to 24.3%. The system becomes more reluctant to compress rather than more willing to make unsafe claims. This asymmetry is desirable: uncertain evidence should cost compression before it costs semantic integrity.

Table 3: Robustness of HARNESSQUOTIENT at budget 16. Semantic signal multiplies the pair specific challenger signal used by both the data generator and the calibrated likelihood model. Observer noise multiplies the default false positive and false negative rates.

Setting	Recall ↑	Unsafe ↓	Safe cov. ↑	Probes ↓
Semantic signal 0.00×	65.9	0.00	0.0	12.8
Semantic signal 0.50×	74.1	0.00	0.0	11.6
Semantic signal 1.00×	88.9	0.92	33.2	8.7
Semantic signal 1.25×	94.5	1.01	66.4	6.1
Observer noise 1×	88.9	0.92	33.2	8.7
Observer noise 5×	88.6	0.84	28.4	8.8
Observer noise 10×	88.0	0.77	24.3	8.8

A.3 CERTIFICATE DIAGNOSTICS

The merge threshold is not a hidden tuning constant. It is the operating point at which a deployment chooses how much unresolved semantic risk it is willing to exchange for interface reduction. Table 4 shows that this choice has a smooth and interpretable consequence. Tightening the threshold from 0.90 to 0.999 drives unsafe merges from 16.1% to zero, but safe merge coverage falls from 97.7% to 13.4%. The system becomes conservative by withholding abstraction, not by changing what counts as a behavioral discrepancy.

Table 4: Risk threshold sensitivity.

τ_m	Unsafe ↓	Safe cov. ↑	Probes ↓
0.900	16.1	97.7	4.6
0.950	11.2	93.2	5.9
0.970	7.8	87.4	7.3
0.980	5.3	80.7	8.6
0.990	2.8	67.9	11.1
0.995	1.2	50.2	13.8
0.999	0.0	13.4	17.7

This distinction is important. If the threshold merely selected a favorable point after test evaluation, the certificate would amount to post hoc tuning. Instead, the main experiment chooses it on validation data against a prespecified unsafe-merge target. The corresponding frontier in Figure 6b is therefore best read as a family of deployment policies. Applications that can tolerate little semantic risk operate toward the conservative end; applications dominated by interface cost may accept a broader quotient. The method exposes that choice rather than hiding it inside a single recommended constant.

A risk threshold is useful only if its confidence carries empirical meaning. Table 5 and Figure 6a examine that question from complementary angles. The Brier score falls from 0.1359 for the passive classifier to 0.0438 for HARNESSQUOTIENT. The improvement does not mean the posterior is a mathematical proof of equivalence. It means that execution evidence, and especially a discovered witness, makes the confidence substantially more aligned with observed outcomes.

Table 5: Equivalence calibration. Lower Brier score is better.

Method	Brier ↓
Passive classifier	0.1359
HarnessQuotient	**0.0438**

The remaining mid-range miscalibration is also instructive. Behavioral equivalence is a statement about contexts that may not yet have been tested, so confidence should be interpreted as an empirical decision statistic tied to the stated context model and observer. We therefore validate calibration and unsafe-merge risk separately. This keeps the certificate honest about what it establishes: not universal identity, but sufficiently low residual risk under the evaluation distribution.

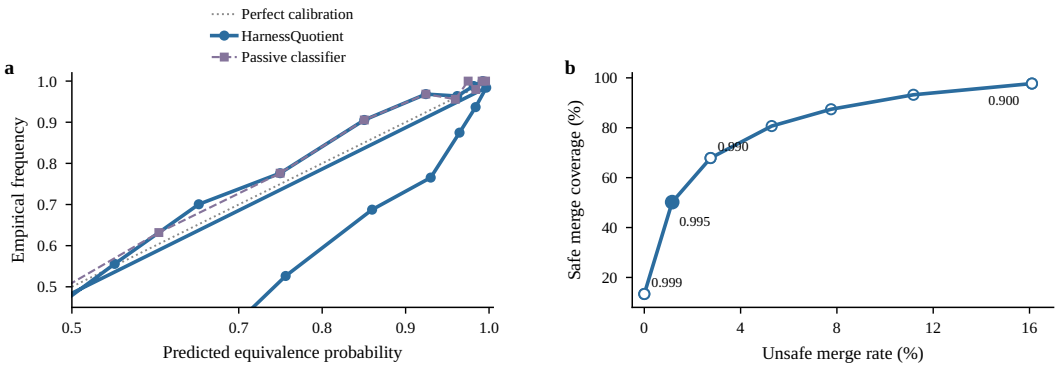


Figure 6: Certificate diagnostics. **a**, calibration of equivalence confidence. **b**, the safety compression frontier induced by the merge threshold.

A.4 Generalization and Component Ablations

The semantic challenger is developed on a subset of distinction families and then evaluated on complete held-out families. Table 6 shows a modest but visible shift: HarnessQuotient moves from 89.8% witness recall on seen families to 85.8% on unseen ones. That four-point loss is worth reporting because it rules out the strongest version of the claim. Semantic guidance is not invariant to the form of a distinction.

Table 6: Witness recall on seen and held out distinction families.

Method	Seen	Held out
IID testing	7.5	7.9
Generic coverage	65.2	64.9
HarnessQuotient	**89.8**	**85.8**

What survives the shift is more informative than what is lost. Generic coverage remains near 65% on both splits, while HarnessQuotient preserves a margin of roughly twenty points on the held-out families. The challenger therefore appears to transfer a useful prior over where behavioral differences tend to hide rather than simply memorizing a particular mutation template. This is the form of generalization that matters for an evolving harness, where the next distinction is unlikely to look exactly like the previous one.

The ablations then separate the mechanisms responsible for that behavior. Removing certification increases unsafe merges from 0.76% to 11.42%, showing that search alone is not a safety mechanism. Removing the semantic challenger reduces witness recall from 87.8% to 65.4%, almost exactly the generic-coverage regime. Replacing active selection with random selection reaches 82.0%, which places the two contributions in order: semantic hypotheses provide the larger gain, and planning extracts additional value from them. Complete linkage addresses a different failure mode. When pairwise decisions are lifted into larger classes, single linkage allows local mistakes to propagate through a chain; contamination rises from 1.97% to 9.18%.

Table 7: Measured mechanism ablations. Removing certification sharply increases unsafe merges, while replacing active selection lowers witness recall.

Variant	Recall ↑	Unsafe ↓	Safe cov. ↑	Probes ↓
Full	87.8	0.76	35.4	8.9
No certification	88.6	11.42	98.7	10.1
No semantic challenger	65.4	0.00	0.0	12.8
Noisy observer	87.9	0.64	30.4	8.9
Random selection	82.0	0.36	20.3	10.4
Complete linkage contamination			1.97%	
Single linkage contamination			9.18%	

A.5 Scaling and Continual Evolution

Scaling tests a different failure mode from pairwise verification. A quotient can be locally accurate yet cease to be useful when the physical harness grows, either because its interface expands almost as fast as the original library or because small merge errors accumulate into operational risk. We therefore track interface exposure and unsafe behavior together as the capability library grows from 50 to 1,000 entries.

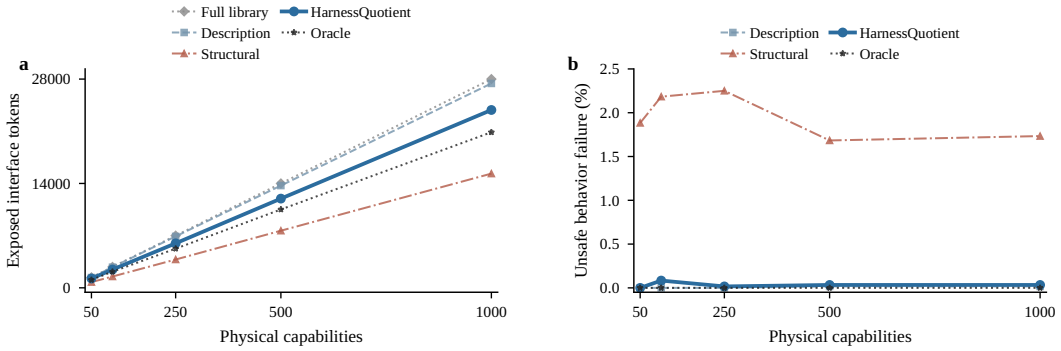


Figure 7: Scaling with the physical capability library. **a**, exposed interface tokens. **b**, unsafe behavior failure. Aggressive structural compression produces the smallest interface, but its semantic errors grow with the library.

Table 8: Scaling to 1,000 physical capabilities. Interface reduction is relative to the full library.

Method	Success ↑	Intf. red. ↑	Sel. err. ↓	Unsafe ↓	Probes ↓
Full library	96.65	0.0	3.35	0.00	0.0
Description similarity	96.70	2.1	3.28	0.02	0.0
Structural contract	96.83	45.3	1.43	1.73	0.0
Generic coverage	96.65	0.0	3.35	0.00	23.2
HarnessQuotient	96.23	14.8	3.73	0.03	13.1
Oracle quotient	95.93	25.5	4.07	0.00	0.0

At 1,000 capabilities, structural matching exposes 15,322 token equivalents, compared with 23,856 for HarnessQuotient and 28,000 for the full library. Its extra compression is not free: unsafe behavior failure reaches 1.73%, while HarnessQuotient remains at 0.03%. The roughly 52 fold gap is easy to miss if one reports only average task success, because an aggressively reduced action space also lowers selection error. The relevant scaling question is therefore not how small the interface becomes in isolation, but how much semantic risk is paid for each unit of compression.

Growth and drift stress different assumptions. Scaling asks whether the quotient remains economical as more capabilities enter the harness; continual evolution asks whether yesterday's equivalence remains valid after behavior changes. A system can pass the first test and fail the second. In particular, task success can stay high while a small set of stale merges accumulates around rarely exercised contexts. We therefore track stale-merge rate directly and inject two semantic changes after earlier merge decisions have already been made. Revalidation is successful only if it reopens the affected boundary without forcing unrelated classes to be rebuilt from scratch.

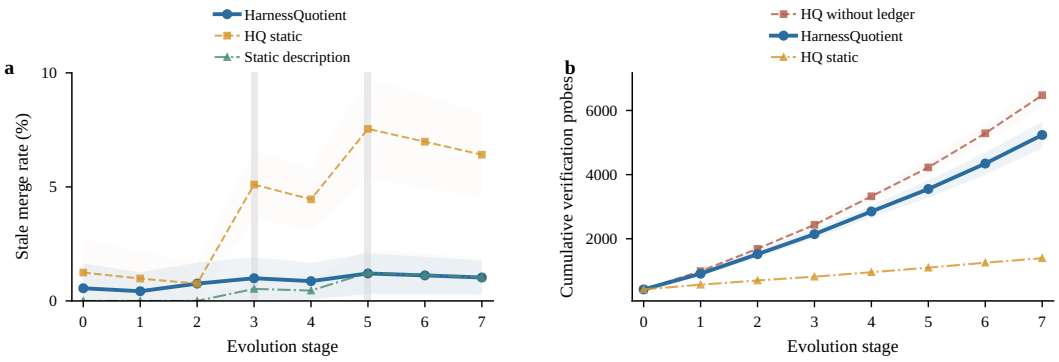


Figure 8: Continual harness evolution. **a**, stale merge rate across two semantic drift events. **b**, cumulative verification probes. Revalidation restores semantic boundaries, while the witness ledger avoids recollecting evidence from scratch.

Table 9: Final stage of the continual evolution stream. Drift detection is measured on capabilities whose behavior changes after an earlier merge.

Method	Retention ↑	Stale ↓	Interface ↓	Cum. probes ↓	Drift det. ↑
No merge	100.00	0.00	400.0	0	0.0
Static description	99.93	1.04	396.0	0	0.0
HQ static	99.57	6.41	377.2	1394	0.0
HQ without ledger	99.95	1.03	384.4	6475	91.5
HarnessQuotient	99.95	1.03	384.6	5236	90.9

A static quotient accumulates stale equivalences when the environment changes. Revalidation restores the boundary, but recurring checks have a cost. The witness ledger reduces cumulative verification by 19.1% without changing final retention or stale merge rate. The useful object produced by counterexample search is therefore not only the current merge decision. It is a compact record of the boundary that justified that decision.

A.6 Rare Context Stress and Amortization

The rare-context stress test asks a deliberately adversarial question: what happens when distinctions that are uncommon in the natural task mixture become operationally important? Table 10 shows that structural compression degrades rapidly under this shift. Increasing rare-context frequency by a factor of four raises its unsafe behavior failure from 0.87% to 6.92%. HarnessQuotient remains at 0.12%.

Table 10: Unsafe failure under more frequent rare contexts.

Rare scale	Structural	**HQ**	Oracle
0.5	0.87	**0.03**	0.00
1.0	1.68	**0.03**	0.00
2.0	3.77	**0.00**	0.00
4.0	6.92	**0.12**	0.00

This result explains why average task success is a weak safety proxy for semantic compression. A method can perform well on common tasks while having erased precisely the distinction that a shifted workload begins to exercise. The appropriate question is therefore not whether compression preserves the mean trajectory under today's mixture, but whether the semantic boundaries that were removed remain behaviorally irrelevant when the mixture changes.

Counterexample search creates an up-front build cost, so the quotient is economically useful only if later tasks repay it. Under the normalized cost model, probe costs of 0.05, 0.10, 0.25, 0.50, and 1.00 break even after 194, 388, 970, 1,940, and 3,879 downstream tasks, respectively. The near-linear scaling matters more than any single break-even number. It makes the economic condition for verification explicit: the more expensive the environment interaction, the longer the horizon over which a certified quotient must be reused.

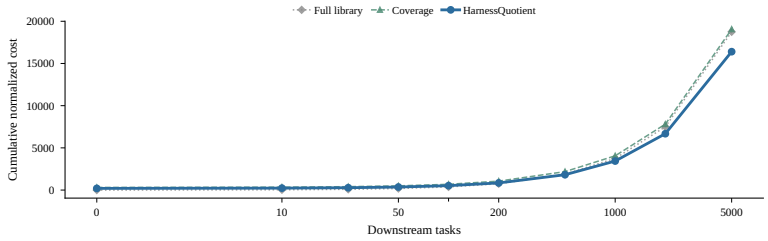


Figure 9: Cumulative normalized cost for a representative verification probe cost of 0.10. The quotient incurs an up front verification cost and repays it only through repeated downstream use.

B Public Benchmark Evaluation Protocol

The controlled benchmark isolates the merge mechanism. The public evaluation asks whether the same distinction-preserving compression remains useful on real agent workloads. We use BFCL (Patil et al., 2025), WorkflowBench (Fan et al., 2025a), and Toolathlon (Li et al., 2026). AgentStudio (Zheng et al., 2025) is reserved as a secondary environment if compute permits.

B.1 Pair Construction Without Label Leakage

The pair benchmark and the end-to-end benchmark serve different purposes. End-to-end evaluation leaves the original public tasks unchanged. Pair evaluation is constructed only from source capabilities that can be replayed from a resettable initial state using an official or deterministic local executor. Non-replayable entries remain in the end-to-end test set but receive no equivalence label.

Equivalent pairs are produced with semantics-preserving adapters such as renaming, argument aliases, optional no-op arguments, canonical response wrappers, and equivalent workflow factoring. Non equivalent pairs preserve common behavior but introduce exactly one hidden distinction from the seven families used in the controlled benchmark. The transform identifier is never exposed to the semantic challenger. Positive transforms must match the source on the source regression suite and a generated boundary suite. Negative transforms must match the source on the common suite and expose at least one reserved separating witness. Pairs with accidental additional differences are rejected.

Source capabilities are split before transformations are created, so all variants of a source remain in the same train, validation, or test partition. We additionally hold out complete distinction families. Task traces used to summarize a capability are disjoint from the hidden witness contexts used for evaluation.

Candidate proposal is evaluated separately from equivalence verification. The mechanism experiments use a fixed candidate set so that verification quality is not confounded by retrieval. The public pipeline uses a shared candidate generator for every merge method and reports candidate recall at several values of k. End-to-end compression is measured using only retrieved candidates. This separates the question of which capabilities should be compared from the question of whether a proposed merge is safe.

Table 11: Public benchmark roles in the final evaluation. Pair labels are created only for replayable capabilities with resettable initial state.

Benchmark	Pair unit	Source-disjoint split key	Pair-level restriction	End-to-end role
BFCL	Function interface	Source function	Replayable local executor	Function calling and multi-turn tasks
WorkflowBench	Workflow program	Application + workflow	Replayable workflow state	Seen- and unseen-API workflows
Toolathlon	Environment tool	Environment + tool	Resettable verified or sandbox executor	Long-horizon verified tasks

B.2 Naturally Occurring Capability Pairs

Controlled transformations provide clean ground truth, but they do not answer how often consequential distinctions arise without intervention. We therefore keep a separate natural-pair subset rather than mixing the two sources. Candidate pairs are mined from the unmodified public capability libraries using the same description, schema, dependency, and trace signals used by the shared proposal stage. No mutation identifier or synthetic guard exists in this subset.

Natural negative pairs are admitted only when an independent offline search produces a separating witness that reproduces from a reset state across repeated executions. The witness is withheld from every evaluated method. Natural positive pairs require stronger provenance than a failure to find a counterexample: they must be documented aliases, deprecated wrappers over the same canonical implementation, or independently verified adapters that dispatch to the same backend semantics. Ambiguous pairs remain unlabeled and are excluded from pairwise accuracy metrics, although they may still appear in end-to-end harness tasks.

We report the natural subset separately from the controlled benchmark. This prevents easy natural duplicates from diluting the hard synthetic negatives, while also preventing synthetic mutations from standing in for evidence that the same failure pattern occurs in real tool libraries. The two evaluations answer different questions: controlled pairs test whether the mechanism can recover a hidden distinction, while natural pairs test whether the mechanism remains useful when the distinction was not authored for the benchmark.

B.3 Matched Information and Cost Budgets

All methods receive the same candidate pairs. Passive methods receive descriptions, traces, schemas, and declared contracts. Active non semantic methods receive the same valid-context generator and environment-execution budget. Semantic random and HarnessQuotient receive the same semantic challenger output, model, token cap, differential observer, and merge certifier. Their only difference is query selection.

The primary verification budget counts differential environment executions because these calls dominate the cost of testing a candidate merge. We additionally report challenger input tokens, challenger output tokens, total model tokens, wall-clock time, and normalized total cost. Public systems are reported in their published native configuration and, when the implementation permits it, in a cost-matched setting with the same backbone and total inference budget.

Table 12: Information available to the controlled merge baselines. Semantic random and HarnessQuotient receive the same challenger output and certifier; only context selection differs.

Method	Static evidence	Semantic hypotheses	Env. execution	Certifier
Description / trace / structural	•			rule-specific
Passive classifier	•	•		learned
IID / fuzz / generic coverage	•		•	shared
Semantic random	•	•	•	shared
HarnessQuotient	•	•	•	shared

Table 12 is deliberately an information budget rather than a taxonomy of algorithms. The strongest controlled comparison is between semantic random and HarnessQuotient, because both receive the same candidate pair, the same semantic hypotheses, the same differential observer, and the same merge certifier. Any difference between them can therefore be attributed to the choice of which valid context to execute next. The non-semantic active baselines answer a different question: how far can one get from coverage alone when pair-specific semantic guidance is removed? Reporting both comparisons prevents gains from semantic information and gains from active planning from being conflated.

Table 13: External comparison set. Each system is evaluated only where its abstraction is meaningful. Public baselines are reported in their native configuration and, when possible, under a matched backbone and total inference budget.

Baseline	Primary role	End-to-end	Continual
Full library / no merge	Uncompressed reference	•	•
Dense retriever	Same-backbone retrieval control	•	
ToolLLM retriever (Qin et al., 2024)	Learned tool retrieval	•	
WorkflowLlama (Fan et al., 2025a)	Workflow specialist	WorkflowBench	
MetaFlowLLM (Fan et al., 2025b)	Experience merging	•	•
SkillZip / ReZip (Tan et al., 2026)	Contract-preserving compression	•	•
HarnessLens (Xu et al., 2026)	Behavior-aware verification	•	•
IsCiL (Lee et al., 2024)	Continual retrievable skills		•
SIL-C (Lee et al., 2025)	Policy-compatible skill learning		•
HarnessQuotient	Behavioral quotienting	•	•
HarnessQuotient + retriever	Orthogonality to retrieval	•	
Oracle quotient	Upper bound	•	•

Table 13 keeps external systems within the regime for which their abstraction is meaningful. Retrieval systems are judged on whether they reduce interface exposure without deleting behavior. Continual-skill systems are judged on retention under growth. Compression systems are judged on both compactness and semantic error. This separation is intentional. A single leaderboard would encourage comparisons between methods that solve different bottlenecks and would obscure whether an observed gain comes from retrieval, memory, compression, or verification.

B.4 Public End-to-End Metrics

The primary public metrics are original benchmark task success, unsafe behavior failure, exposed interface tokens, tool-selection error, environment executions, and normalized total cost. Candidate generation reports recall at k. Pair verification reports witness recall, unsafe merge rate, safe merge coverage, and differential executions. Every benchmark is reported separately and as a macro average. Confidence intervals use paired bootstrap resampling over identical source pairs or tasks.

The metrics must be read jointly. A method that improves task success only by exposing more of the original interface has not demonstrated useful quotienting. A method that reduces the interface while increasing unsafe failure has compressed too aggressively. The desired result is not a single maximum but a better operating point: comparable task success, lower interface exposure, lower selection burden, and no material increase in semantic risk. The retriever combination is especially important because it tests whether quotienting removes redundancy that retrieval alone cannot resolve.

Cross-backbone evaluation guards against a model-specific explanation. The quotienting mechanism should not depend on one model being unusually good at reading tool descriptions or inventing edge cases. We therefore interpret an external result as robust only when the safety-compression tradeoff persists across open backbones of different capacity, even if their absolute task success differs.

The continual comparison is stricter still. Final success is insufficient if the system forgets earlier distinctions or repeatedly rebuilds the same evidence. We therefore report retention, stale-merge rate, interface reduction, and verification cost together. A useful self-evolving harness should reopen certificates after behavioral change, preserve unaffected classes, and reuse prior witnesses so adaptation cost does not grow linearly with system lifetime.

The compiled manuscript intentionally contains no simulated public-benchmark results. Draft layouts for those tables are retained outside the compiled paper and will be populated only by completed runs under the protocol above.

C Formal Properties and Scope of the Certificate

C.1 Why the Verification Objective Is Submodular

For a fixed non-equivalence world θ, define

$$f_\theta(S) = 1 - \prod_{x \in S} (1 - q_\theta(x)) \,. \tag{19}$$

The function is nonnegative and monotone because adding a test can only increase the probability that at least one member of the set exposes the world. Its marginal gain for adding context x to an existing set S is

$$f_\theta(S \cup \{x\}) - f_\theta(S) = q_\theta(x) \prod_{s \in S} (1 - q_\theta(s)) \,. \tag{20}$$

If $A \subseteq B$, then the product term for B is no larger than the product term for A. The marginal value of the same test therefore decreases as the selected set grows. Thus f_θ is submodular. The objective in Equation 11 is a nonnegative weighted sum of these functions, so it is also monotone and submodular. Under a cardinality budget, greedy marginal-gain selection consequently inherits the standard $(1 - 1/e)$ approximation guarantee (Nemhauser et al., 1978).

The result is useful for a practical reason. The semantic challenger does not need to solve a combinatorial optimization problem exactly. It only needs to assign meaningful exposure probabilities to valid contexts. The planner can then use a simple greedy procedure with a known worst-case guarantee, and the full system can spend its complexity budget on constructing good behavioral hypotheses rather than on expensive search over test subsets.

C.2 What a Certificate Does and Does Not Establish

A merge certificate is deliberately narrower than a proof of program equivalence. It states that, under the modeled context family, observation function, observer error rates, and collected evidence, the posterior probability of a material distinction has fallen below the deployment threshold. Four boundaries follow immediately.

First, a distinction outside the context generator cannot be ruled out. Second, an effect that is invisible to Ω cannot be detected even if the corresponding context is tested. Third, a misspecified likelihood model can make posterior risk overconfident. Fourth, candidate generation can prevent two genuinely redundant capabilities from ever reaching verification. These are not implementation details. They define the scope of the claim.

The design responds to these limits asymmetrically. Missing evidence should primarily reduce compression by increasing abstention. Discovered counterexamples immediately block merging. Observer uncertainty is folded into the likelihood rather than treated as noise-free truth. Changes to implementations or dependencies invalidate affected certificates. None of these mechanisms can convert an incomplete observation model into universal semantic knowledge. Their purpose is to make the remaining uncertainty visible and operational.

D Reproducibility Details

D.1 Code and Artifact Availability

The anonymous implementation and experiment artifacts accompanying this submission are available at

`https://anonymous.4open.science/r/HARNESSQUOTIENT-55BC`.

The repository is intended to make the evaluation path inspectable rather than merely executable. It contains the experiment configuration, implementation, plotting pipeline, and the scripts needed to reproduce the controlled studies reported in this paper. We keep the repository anonymous for double blind review and use the same configuration files to define the reported budgets, random seeds, observer parameters, and merge thresholds.

Table 14: Failure modes, their consequences, and the diagnostic intended to expose them. The table separates failures that mainly reduce compression from failures that can threaten semantic safety.

Source	Consequence	Default response	Diagnostic
Candidate miss	A redundant pair is never compared	Preserve both capabilities	Candidate recall@k
Challenger miss	Important contexts receive too little probability mass	Fall back toward generic coverage; abstain if evidence remains weak	Held-out families; signal ablation
Observer false negative	A tested distinction may remain unseen	Use calibrated likelihood and repeated or downstream probes	Noise sensitivity; calibration
Observer blind spot	A relevant effect is absent from Ω	Extend the observation signature before certifying the pair	Delayed-effect and verifier-blind-spot families
Distribution shift	A previously harmless distinction becomes important	Invalidate affected certificates and replay witnesses	Rare-context stress; continual drift
Linkage chain	Local pairwise errors propagate into a larger class	Require complete-linkage admission	Class-contamination ablation

Table 15: Core controlled-evaluation settings. All values are read from the released YAML configuration.

Setting	Value
Context factors	12
Max. active factors	4
Natural activation probability	0.08
Hidden guard sizes	2, 3, 4
Distinction families	7
Observer false positive	0.001
Observer false negative	0.015
Equivalence prior	0.80
Default merge threshold	0.995
Minimum tests before merge	4
Unsafe-merge validation target	2%
Main seeds	101, 103, 107, 109, 113

D.2 Controlled Benchmark Configuration

The main controlled benchmark is generated entirely from YAML configuration. It uses twelve binary context factors, at most four simultaneously active factors, and a natural activation probability of 0.08 per factor. Non-equivalent pairs use hidden guard sizes two, three, and four with probabilities 0.55, 0.35, and 0.10. The benchmark contains seven distinction families: rare preconditions, state effects, delayed effects, error behavior, order sensitivity, schema edges, and verifier blind spots. The default differential observer has false-positive probability 0.001 and false-negative probability 0.015.

The equivalence prior is 0.80. The default merge posterior threshold is 0.995, with at least four tests required before a merge can be certified. Threshold selection uses validation data only and targets an unsafe-merge rate of two percent. The primary witness study uses 500 validation pairs and 1,200 test pairs at budget 32 with seeds 101, 103, 107, 109, and 113. The matched budget frontier uses 1,000 test pairs at budgets 4, 8, 16, and 32 with the same five seeds. Reported uncertainty is computed across seeds; paired bootstrap analyses use 10,000 resamples where stated.

D.3 Interpreting the Synthetic Benchmark

The controlled benchmark is not intended to stand in for the full diversity of real software tools. Its role is narrower: it creates executable pairs for which the hidden semantic distinction is known, while withholding that distinction from the challenger. This makes it possible to measure false merges, witness recall, calibration, and sample efficiency without relying on an LLM judge for ground truth.

The benchmark is therefore strongest as a mechanism test. It can show whether semantic guidance helps locate rare distinctions, whether active selection uses a fixed budget efficiently, and whether the certifier reacts sensibly to noise and drift. It cannot by itself establish how often such distinctions arise in natural harnesses. That question belongs to the public benchmark evaluation and to naturally occurring capability pairs. We keep those two claims separate throughout the paper.

D.4 Qualitative Witness Archetypes

The seven distinction families correspond to recurring ways in which apparent interchangeability can fail. A *rare precondition* pair agrees until a conjunction of state flags activates a guarded branch. A *state effect* pair returns the same value but mutates external state differently. A *delayed effect* pair is indistinguishable at the immediate call boundary but changes a later read or action. An *error behavior* pair agrees on nominal inputs but diverges on malformed or partially valid arguments. An *order-sensitivity* pair differs only when another operation has occurred before it. A *schema-edge* pair separates on a boundary representation such as a missing field or structurally unusual record. A *verifier blind spot* pair passes the immediate checker while leaving a downstream state that later behaves differently.

These archetypes matter because they defeat different shortcuts. Surface similarity misses all of them. A purely structural contract can detect some schema differences but not delayed state. Immediate output comparison can miss both state effects and verifier blind spots. The purpose of the distinction taxonomy is therefore not to enumerate every possible failure, but to force the evaluation to include failures that live at different layers of the harness.

E Complexity, Persisted State, and Auditability

E.1 Verification Cost and Class Growth

The expensive unit in HarnessQuotient is not posterior arithmetic but differential execution. Let M denote the number of candidate pairs proposed in one evolution stage and let B be the maximum verification budget per pair. Pair verification therefore uses at most MB matched executions before early stopping. In practice the realized cost is lower for two reasons. A separating witness terminates a pair as soon as a material difference is observed, while sufficiently strong negative evidence can certify a merge before the nominal budget is exhausted. This is why we report both the budget and the realized mean number of probes.

Class construction adds a second source of cost. Complete-linkage admission requires a new capability to be supported against every existing member of the target semantic class rather than only against one representative. For a class of size m, the worst-case admission cost is therefore proportional to mB. This conservative rule is intentional because single-linkage admission converts local pairwise errors into class-wide contamination. The witness ledger mitigates the practical cost: stored witnesses are replayed first, so many invalid admissions can be rejected without launching a fresh search. The continual experiment measures this reuse directly.

The resulting complexity has a useful interpretation. HarnessQuotient does not make large harnesses free. It moves cost from every downstream decision into occasional evidence gathering at abstraction boundaries. The method is attractive when a quotient is reused often enough that this up-front verification cost can be amortized. For small or short-lived harnesses, retrieval without merging can remain the better engineering choice.

E.2 Persisted Evidence and Decision Audit Trail

A quotient entry is accompanied by the evidence that made the abstraction defensible. We persist three kinds of records: candidate metadata, separating witnesses, and merge certificates. Candidate metadata records why two capabilities were compared but carries no authority to merge them. A separating witness records a resettable context and the observed behavioral discrepancy; it is durable negative evidence and can be replayed after future harness changes. A merge certificate records the context model, observation signature, observer calibration, tests executed, posterior state, threshold, and dependency versions that were in force when the merge was admitted.

Table 16: Persisted state used to audit and revalidate quotient decisions. Proposal evidence and decision evidence are intentionally stored separately.

Record	Stored fields	Role
Candidate metadata	Pair identifiers, proposal scores, schemas, trace summaries, dependency versions	Explains why the pair entered verification; never sufficient for merging
Separating witness	Reset state, context, matched executions, observed difference, verifier output	Blocks unsafe merges and becomes a regression test after harness change
Merge certificate	Context model, observation signature, observer error model, executed tests, posterior, threshold, version scope	Documents the evidence and assumptions under which interchangeability was accepted
Class membership	Member identifiers, supporting pair certificates, invalidation dependencies	Prevents unsupported transitive admission and enables targeted revalidation

This separation is important for auditability. A semantic description can explain why two tools looked redundant, but it cannot be retroactively treated as evidence that they were interchangeable. Conversely, a witness remains useful even if the model that originally proposed the distinction is replaced. The durable artifact is the executable boundary, not the language model's explanation of it.

E.3 Invalidation and Revalidation Semantics

Certificates are version-scoped. A change to a capability implementation, declared dependency, observation signature, verifier, or context generator invalidates only the certificates whose recorded scope intersects that change. Revalidation begins with stored witnesses, then replays previously informative contexts, and only then allocates new search budget. This ordering preserves two properties that matter in a continually evolving harness: previously discovered failures are cheap to rediscover, and unrelated classes do not need to be reopened merely because some other component changed.

The invalidation policy is deliberately conservative about missing provenance. If the system cannot determine whether a change affects a certificate's assumptions, the certificate is reopened rather than silently preserved. This may temporarily increase interface size, but it makes uncertainty visible as abstention instead of converting provenance gaps into semantic claims.

F Limitations and Intended Scope

The method is most appropriate when candidate capabilities can be replayed from matched or sufficiently controlled state. Highly nondeterministic external services, irreversible side effects, and environments without reliable reset semantics make differential execution substantially harder. In those settings, the observation model must include uncertainty induced by the environment itself, and a larger fraction of candidate pairs should remain separate.

The method also introduces verification cost before it returns interface savings. The amortization study makes this tradeoff explicit: expensive environments require a longer downstream horizon before the quotient repays its construction cost. For short-lived harnesses with small capability sets, no-merge or retrieval-only systems may be the better engineering choice.

A second limitation concerns external validity. The controlled benchmark deliberately supplies known latent distinctions so false merges can be measured without an LLM judge. That design isolates mechanism quality but does not estimate how frequently such distinctions occur in natural software ecosystems. The natural-pair subset and public benchmark protocol are therefore essential to the final empirical claim. A strong controlled result with weak natural-pair evidence would support the verification mechanism, but not the prevalence or practical value of quotienting in deployed harnesses.

Finally, the semantic challenger is a source of inductive bias rather than an oracle. Better models can improve the ranking of plausible distinctions, but a confident challenger can still be wrong. The central safety mechanism is therefore not trust in the challenger. It is the separation between proposing a distinction and certifying a merge, together with abstention when evidence is insufficient.

The strongest version of the method's claim is correspondingly modest: active behavioral testing can make semantic compression more evidence-efficient and easier to maintain. It cannot make unobserved behavior disappear.

F.1 Operational Preconditions and Stop Conditions

A practical deployment should enable quotienting only when the evidence pipeline is itself inspectable. The most important preconditions are not model capacity or prompt quality. They are control over state, visibility into effects, and the ability to invalidate stale evidence. The summary below states the conditions under which a merge certificate should be considered actionable.

Table 17: Operational preconditions for using a behavioral quotient. A failed precondition should reduce compression or suspend merging rather than be hidden by a lower threshold.

Precondition	Evidence required	Stop condition
Matched or resettable state	Both capabilities can be replayed from a common initial state with controlled exogenous inputs	Preserve both capabilities if state cannot be aligned well enough for a meaningful differential comparison
Observable material effects	The observation signature includes outputs, state changes, side effects, and delayed checks relevant to downstream use	Reopen the observation model before certifying if a material effect can occur outside the recorded signature
Calibrated differential observer	Error rates are estimated on held-out comparisons and incorporated into the certificate likelihood	Increase abstention or suspend merging when observer drift invalidates the calibration model
Traceable dependency versions	Implementations, schemas, verifiers, and relevant dependencies carry version provenance	Invalidate every certificate whose recorded scope intersects an untracked or incompatible change
Sufficient reuse horizon	Expected downstream reuse can plausibly amortize the up-front verification cost	Prefer retrieval or no merge when the harness is short lived or rarely reused

These conditions turn abstention into an engineering contract rather than a residual category. When any prerequisite fails, the conservative response is not to lower the threshold until a merge becomes possible. It is to preserve the distinction until the missing source of evidence can be restored. This is the operational counterpart of the paper's central principle: compression is justified by evidence that a distinction no longer matters, not by pressure to make the interface smaller.

F.2 Reporting Discipline for External Evaluation

External results should preserve the same separation of questions used throughout the paper. Candidate recall, pair-verification quality, end-to-end task success, and total cost should be reported as distinct quantities rather than collapsed into one score. Controlled transformed pairs and naturally occurring pairs should also remain separate. Pooling them would allow abundant easy aliases to mask poor performance on consequential distinctions, or allow deliberately difficult mutations to exaggerate the prevalence of the problem in natural libraries.

Coverage must be reported together with accuracy. In particular, the final evaluation should state what fraction of capabilities are resettable enough for matched differential execution, what fraction enter the candidate graph, and what fraction ultimately receive a merge decision. Methods should share the same candidate pool whenever the scientific question is verification, and the same backbone, token accounting, environment-execution budget, and latency measurement whenever the question is end-to-end cost. A result is persuasive only when its operating point remains favorable under those matched conditions, not when compression is purchased by silently discarding difficult capabilities from evaluation.