INSPECT OR ROLL BACK? ASYMMETRIC DIAGNOSTICS FOR CHECKPOINTED RECOVERY

Anonymous authors
Paper under double-blind review

ABSTRACT

When a stateful workflow fails, recent checkpoints may already contain the mistake. A controller can inspect an earlier prefix or discard it and repeat the remaining work. We study how the two kinds of inspection error should affect this choice: falsely trusting a bad prefix risks an incorrect commit, whereas falsely rejecting a good prefix wastes computation. An ordered recovery model separates these consequences and supports inexpensive asymmetric diagnostics. A three-cost representation accounts exactly for mistaken execution paths within a restricted policy class; we extend it to failure penalties and persistent hidden error modes. Experiments compare calibrated unit-test services, shallow lookahead, an edge-cutting baseline, and broader-history policies. In an executed multi-file code-edit workflow, asymmetric tests reduce the one-step controller's checking and replay work by 18%; two-step lookahead matches full compilation, and repairing the test oracle saves substantially more. Risk-matched comparisons and measured planning costs identify full compilation as a conditional refinement for costly, reused workflows. The main practical lesson is to improve and calibrate the diagnostic before using deeper recovery planning.

1 INTRODUCTION

A workflow edits several files, updates a cache of assumptions, and then fails a regression check. The latest checkpoint may preserve the same mistaken edit. Full restart removes the mistake but discards valid work. Selective recovery instead asks which prefix is worth trusting. This problem arises around stateful tool execution, independently of how the original actions were generated. Reasoning–action systems and persistent agent memories motivate such state reuse (Yao et al., 2023; Shinn et al., 2023; Packer et al., 2023); our experiments concern executed structured workflows and authored code mutations.

Inspection errors have different consequences. A false-clean report may retain corruption. A false-bad report can cause safe but unnecessarily deep rollback. Requiring both probabilities to be tiny can therefore require evidence that the recovery objective does not need. Conversely, cheaply allowing false-bad reports makes their downstream cost important: a planner that treats every report as correct can underestimate repeated work. We separate *diagnostic design*, which any controller can use, from *planning depth*, which adds computational overhead.

The main empirical effect is diagnostic design. In a controlled execution grid, switching the one-step policy from symmetric to asymmetric services explains 74% of its total cost reduction to the full compiler. In the code-edit case, shallow planning captures the full compiler's gain, while correcting a brittle unit-test oracle is substantially better than sophisticated planning around it. These findings motivate a hierarchy: improve the checker, calibrate its error directions, try inexpensive lookahead, and compile only when residual savings repay its cost.

The theory explains this hierarchy. First, a first-error argument identifies which diagnostic errors consume the safety budget, including a bound for episode-persistent common-mode failures. Second, an ordered interval has only three relevant execution-cost coordinates: the state lies inside, earlier, or later. This yields exact actual-noise optimization in the chosen class, an additive approximation, and a numerical evaluation of an instance-specific optimality bound. Third, shallow, risk-matched, and broader-history comparisons show the cost of both the algorithm and its restrictions. Decision-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

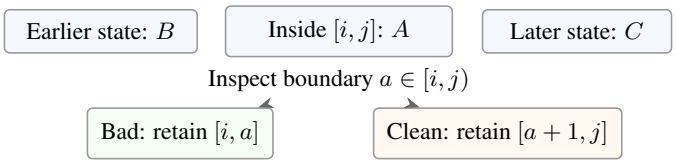

Figure 1: Three ways to reach a recovery interval. A mistaken report can put the true state outside the retained interval. The compiler keeps both outside costs instead of treating the current interval as certain. A false-bad departure is conservative; a false-clean departure is unsafe.

oriented diagnosis is established (Breese & Heckerman, 1996; Golovin et al., 2010; Javdani et al., 2014); the specialization here is asymmetric checkpoint recovery and its cost accounting.

2 Checkpoints, diagnostics, and the recovery objective

There are $n - 1$ completed stages. A hidden state $J \in \{1, \dots, n\}$ denotes the first persistent fault; $J = n$ means the completed state is clean. Checkpoint $K \in \{0, \dots, n - 1\}$ is safe exactly when $K < J$. Restoring it and regenerating the suffix costs

$$R_0 \geq R_1 \geq \cdots \geq R_{n-1} \geq 0. \tag{1}$$

Checkpoint zero is trusted. Restoration includes memory and workspace, and regeneration uses a trusted source. The already-executed prefix is a common sunk cost. Every policy terminates with a recovery action; unsafe completion and unnecessary replay are distinct outcomes.

A diagnostic service s at boundary a reports whether $J \leq a$. Write

$$\alpha_s = \mathbb{P}(\text{clean} \mid J \leq a), \quad \beta_s = \mathbb{P}(\text{bad} \mid J > a), \quad t_s^b, t_s^g = \text{conditional mean service costs.} \tag{2}$$

The cost includes every raw query. A service's duration and report may be correlated. For the basic compiler, conditional service laws depend on the comparison truth and chosen service, and future observations are fresh. Section 3.1 treats persistent hidden modes. The prior b_j guides expenditure; safety is statewise. We study

$$\min_{\pi} F_{b,\lambda}(\pi) = \mathbb{E}_b\left[\sum_s C_s + R_K\right] + \lambda \mathbb{P}_b(K \geq J), \qquad \max_j \mathbb{P}_j(K \geq j) \leq \delta. \tag{3}$$

Here $\lambda \geq 0$ prices the downstream consequence of an incorrect commit, separately from the executed work. The default $\lambda = 0$ is resource minimization under a risk cap, not an assertion that failures are free. Section 6 varies both parameters.

The ordered policy class. At an interval $[i, j]$, a policy either stops at checkpoint $i - 1$ or inspects $i \leq a < j$. A bad report selects $[i, a]$; a clean report selects $[a + 1, j]$. At most D services are completed. The class $\mathcal{T}_D(\mathcal{S})$ fixes a finite service menu and permits report-dependent continuation. It includes restart. It excludes revisiting boundaries, early interruption of a service, and arbitrary terminal checkpoints. The broader-history reference in Section 6.1 relaxes these choices.

Why ordinary retry is a separate option. If memory is persistently wrong with probability ρ, and clean executions fail transiently with probability p, then r same-memory retries succeed with probability $(1 - \rho)(1 - p^r)$. The ceiling $1 - \rho$ follows directly from conditioning on memory validity. Retrying can be valuable, but it does not remove a persistent fault in this model. We execute a retry baseline in the code study; the compiler chooses between inspection and rollback.

3 Spend evidence on the dangerous error

Theorem 1 (Directional safety)**.** *Fix a state and an ordered recovery tree. If all reports so far are correct, the retained interval contains the state. Its first false-clean report makes every descendant*

recovery unsafe; its first false-bad report makes every descendant recovery safe. With fresh services,

$$r_j = \sum_{v \in P_j : j \le a_v} \alpha_{s_v} \prod_{u \prec v,\, u \in P_j} (1 - e_{j,u}), \quad e_{j,u} = \begin{cases} \alpha_{s_u} & j \le a_u, \\ \beta_{s_u} & j > a_u. \end{cases} \tag{4}$$

P_j is the truthful path. More generally, conditional-on-history false-clean bounds $\bar{\alpha}_v$ imply $r_j \le \sum_{v \in P_j : j \le a_v} \bar{\alpha}_v$. In particular, $\bar{\alpha}_v \le \delta / D$ suffices.

This theorem explains the useful asymmetry: β affects cost, but needs no separate cap for safety. Its local allocation is sufficient and can be conservative. Global-risk and broader-history policies provide empirical references for that restriction.

Paid asymmetric tests. For a raw binary-symmetric checker with noise $0 < \eta < 1/2$, add $+1$ for each bad report and -1 for each clean report. Stop at scores $-l$ or $+u$, where l, u are positive integers. Put $z = \eta/(1-\eta)$. Classical stopped-walk calculations (Wald, 1945; Wald & Wolfowitz, 1948) give

$$\alpha = \frac{z^l - z^{l+u}}{1 - z^{l+u}}, \qquad \beta = \frac{z^u - z^{l+u}}{1 - z^{l+u}},$$
$$\mu_b = \frac{u - (l+u)\alpha}{1 - 2\eta}, \qquad \mu_g = \frac{l - (l+u)\beta}{1 - 2\eta}. \tag{5}$$

With $\eta = 0.25$, $l = 6, u = 1$ gives a false-bad probability close to one third while making false-clean small. Requiring evidence on both sides spends more queries. The menu includes a symmetric service and asymmetric upper thresholds one or two, each with the smallest l meeting the same false-clean cap. Every planner pays the same per-query price.

Where calibration labels come from. For a general checker, collect predeclared, independently selected reference prefixes with an externally established good/bad label. Use simultaneous one-sided binomial confidence bounds (Clopper & Pearson, 1934) to bound false-clean probabilities for the complete candidate menu. On their joint coverage event, any subsequently chosen tree inherits Theorem 1. A confidence failure probability ξ adds at most ξ to unconditional deployment risk. Labels cannot be supplied by the uncertain memory being checked. Our code experiment obtains them from full finite-domain contract tests on seeded clean and faulty prefixes. Faults found selectively in deployment are a different sampling distribution.

3.1 Persistent errors and what repeated checks can guarantee

Independence is a property of a diagnostic mechanism, not a benefit of asking the same question again. Suppose the whole interaction law for each state is

$$P_j = (1 - \gamma_j) P_j^{\mathrm{fresh}} + \gamma_j Q_j, \qquad 0 \le \gamma_j \le \bar{\gamma}, \tag{6}$$

where Q_j can be any correlated, even adversarial, transcript law. The mixing event is episode-persistent, rather than resampled per query.

Proposition 2 (Common-mode safety)**.** *If the fresh-channel risk is at most δ_0 statewise, actual risk is at most $\bar{\gamma} + (1 - \bar{\gamma})\delta_0$. This bound is sharp. A δ target is achieved by $\delta_0 \le (\delta - \bar{\gamma})/(1 - \bar{\gamma})$ when $\bar{\gamma} < \delta$. Alternatively, independently restart with probability*

$$q \ge \max\{0, 1 - \delta / [\bar{\gamma} + (1 - \bar{\gamma})\delta_0]\}.$$

In the specified all-clean blind-spot experiment, a 2.5% common-mode probability raises the maximum unprotected risk to 3.13%. The restart gate returns it to at most 1%, requiring up to 68.1% restart probability across the grid. A valid bound on the common-mode probability is a separate requirement. Marginal checker accuracy alone does not identify it. If the persistent mode instead has a known finite conditional model, the compiler can account for it directly using three costs per mode (Corollary 4).

4 PLANNING BEYOND ONE STEP

For a subtree on $I = [i, j]$, let A be its expected remaining objective under the prior restricted to I, B its objective when the true state is earlier than I, and C when the state is later. All earlier states make every interior comparison bad; all later states make every comparison good. Given that the state is inside I, ancestor reports have the same likelihood for each state there. The original prior restricted to I is therefore sufficient.

A leaf has vector $(R_{i-1}, R_{i-1} + \lambda, R_{i-1})$: only an earlier state makes that action unsafe. Let p be the inside prior mass in the left child. A service combines child vectors by

$$A = p[t^b + (1-\alpha)A_L + \alpha B_R] + (1-p)[t^g + \beta C_L + (1-\beta)A_R],$$
$$B = t^b + (1-\alpha)B_L + \alpha B_R, \qquad C = t^g + \beta C_L + (1-\beta)C_R. \tag{7}$$

Figure 1 shows these outside states. The βC_L term charges conservative mistakes; αB_R includes both wasted work and the failure penalty.

Theorem 3 (Actual-noise compilation and approximation)**.** *Enumerating leaf/split/service/child choices and retaining componentwise nondominated triples gives the exact minimizer of $F_{b,\lambda}$ over $\mathcal{T}_D(\mathcal{S})$. If every service meets $\alpha \le \delta/D$, every retained tree is safe. For $h > 0$, bucket the two tail costs and retain the actual tree with smallest A in each bucket. Its objective exceeds the exact class optimum by at most Dh. An optimistic recurrence using lower tail-bucket edges gives*

$$L_h \le F^*_{b,\lambda,\mathcal{T}_D} \le U, \tag{8}$$

where U is the evaluated objective of any feasible incumbent.

Every coefficient in Equation (7) is nonnegative and each row's child weights sum to one. A parent therefore preserves dominance, while bucket error accumulates by at most h per level. The full proofs include zero-mass intervals and the optimistic recurrence. These are real-arithmetic theorems. **Our reported brackets are ordinary-double evaluations, not outward-rounded machine certificates.**

Corollary 4 (Finite persistent modes)**.** *Suppose a hidden mode $M \in \{1, \dots, m\}$ is sampled once per episode. Conditional on (J, M), service observations are fresh, and their laws depend on M and prefix truth. With known $\mathbb{P}(M)$ and $b^{(M)}$, one triple per mode suffices. Combining each mode by Equation* (7)*, retaining nondominated $3m$-vectors, and minimizing $\sum_m \mathbb{P}(M = m)A_m$ is exact within the same ordered class for the joint model.*

The executable tree is shared across modes and never observes M. This provides cost accounting for known common-mode channels, while Proposition 2 covers a bounded but otherwise unspecified contaminated mode. The $3m$-vector frontier can be expensive; the experiments use it as a small exact reference.

A cheaper choice: actual-probability lookahead. Depth-h lookahead evaluates every available split and service for h further decisions, terminates the truncated tree with conservative rollback, executes its first decision, and replans. Posterior weights include states on mistaken branches. Depth one is the myopic comparator; depths two and three provide the key intermediate comparisons. Full compilation may reuse subproblems across these decisions, but it also maintains competing tail-cost vectors. We measure both costs rather than assuming either method is inexpensive.

5 COMPARING DIAGNOSTIC QUALITY AND PLANNING DEPTH

A paired execution protocol. The primary grid has $n = 16, D = 5$, two replay profiles (uniform and clustered, both with restart cost 101), clean-state prior mass 0.2 or 0.8, and raw checker noise 0.1 or 0.25: eight conditions. Raw inspection cost is one and the initial risk cap is 0.01. Each method executes the same 1,000 seeded tasks and raw observation streams per condition. Each study uses distinct seed identifiers; this grid uses new random streams. The executor modifies a symbolic workspace and restores both cached values and dependent artifacts. Every query and replay action is charged. This environment tests the specified probabilistic model; Section 7 executes source files and unit tests.

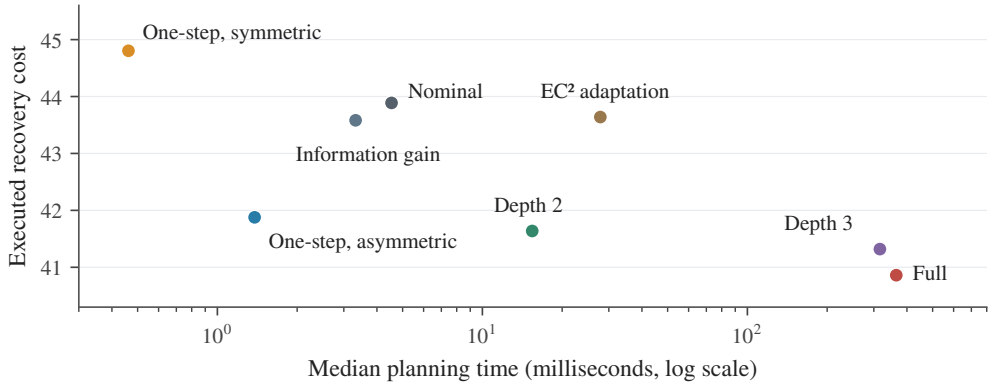

Figure 2: Execution cost and planning time. EC^2 denotes Equivalence Class Edge Cutting. All asymmetric methods use the same menu. Points show actual executed cost against median end-to-end planning time over eight conditions and three timing repetitions. Full compilation offers a small incremental gain; depth-three lookahead is nearly as costly to construct in this implementation.

Table 1: Paired primary executions: 8,000 per controller. Cost counts queries and replay on every episode, including incorrect commits. Planning time, in milliseconds (ms), is separate from execution cost. Error counts alone do not establish risk equivalence.

Controller	Cost	Errors	Plan (ms)
One-step, symmetric	44.80	12	0.46
One-step, asymmetric	41.88	12	1.38
Depth-2 lookahead	41.64	13	15.42
Depth-3 lookahead	41.32	14	316.25
Full compiler	40.86	14	364.58
Nominal compiler	43.89	10	4.54
EC^2 adaptation	43.64	12	27.87
Information gain	43.58	15	3.32
Restart	101.00	0	0.49

Baselines. All asymmetric methods receive exactly the same service menu. The nominal compiler optimizes truthful paths with the same conditional service costs; it is an ablation. The Equivalence Class Edge Cutting (EC^2) adaptation uses expected cross-class edge cutting (Golovin et al., 2010) on 2,048 sampled state/noise hypotheses. We select among five declared recovery-group resolutions by known-model cost, never execution outcomes, and include the search cost in planning time. An information-gain baseline values posterior entropy reduction per paid query. Both use the ordered, conservative-leaf safety wrapper; the original identification approximation guarantees do not transfer to this modified stopping objective. These are explicit adaptations, not implementations of every decision-region or optimal noisy-search algorithm cited below.

Switching the one-step service menu reduces cost from 44.80 to 41.88; full compilation then reduces it to 40.86 (Table 1). The menu therefore explains 74% of the total absolute saving. The full/one-step paired difference is 1.02 units with a stratified 95% Monte Carlo interval $[0.86, 1.18]$. Depth two captures 0.24 units of that difference and takes 15.4 ms to plan, against 364.6 ms for full compilation. Depth three reaches 41.32 at 316.3 ms. These timings construct all reachable replanning nodes; lazy execution can avoid constructing unvisited continuations. The nominal comparison isolates erroneous-branch accounting, but is less practically important than the shallow comparisons. Figure 2 places all of them on the same time–cost plane.

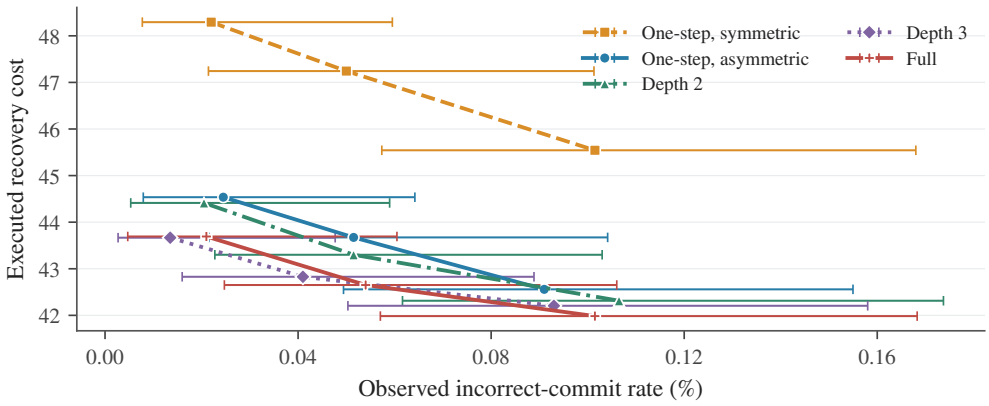

Figure 3: Matched-model-risk policies, independently executed. Each point represents 200,000 raw-score-walk recoveries; horizontal bars are conservative 95% stratified binomial intervals. Realized risks fluctuate around the common targets. The method weights are fixed before these outcomes are observed.

6 Risk, policy restrictions, and compilation economics

Match risk before interpreting small gains. We sweep $\delta \in \{10^{-3}, 3.00 \times 10^{-3}, 0.01, 0.03, 0.1\}$ for symmetric one-step, asymmetric one-step, depth two, depth three, and full compilation: 200,000 workspace executions. We then fit policy-mixture weights using *analytic* costs and risks at caps at most 0.01, plus restart, for marginal risk targets 2.00×10^{-4}, 5.00×10^{-4}, and 10^{-3} under the fixed eight-condition distribution. Evaluation uses new streams and 200,000 score-walk executions per method/target, drawing every raw score increment. All components retain a statewise one-percent cap. This separates calibration/selection from the observed errors used for comparison.

At the 10^{-3} target, exact model costs are 42.68 for one-step, 42.50 for depth two, 42.13 for depth three, and 42.00 for full compilation. Full compilation's matched-risk gain is therefore 1.59% over one-step and 0.30% over depth three. Independent execution yields costs 42.56, 42.31, 42.21, and 41.98, with 182, 213, 186, and 203 errors in 200,000 trials, respectively. Figure 3 reports uncertainty rather than describing unequal realized counts as equal risk. Part of the unadjusted advantage was expenditure of available risk; the main conclusion survives in a smaller form.

Charge for incorrect commits. For $\lambda \in \{0, 100, 10^3, 10^4, 10^5\}$, we optimize Equation (3) with the same uniform cap. At $\lambda = 10^3$, full/asymmetric and one-step/asymmetric objectives are 42.73 and 43.38. At 10^4 they are 52.04 and 52.38; depth two reaches 52.10. At 10^5, depth two and full compilation select equal objective values (77.95). The $(-6, +1)$ service remains selected in some full policies through 10^4, and disappears at 10^5. Asymmetric testing remains useful at moderate penalties; conservative choices become competitive when an incorrect commit is sufficiently costly.

6.1 What the ordered restriction costs at deployment size

A point-based value iteration (PBVI) reference (Pineau et al., 2003) uses $n = 16, D = 5$, 256 belief points, two belief seeds, and five failure penalties. It permits any terminal checkpoint and repeated boundaries. We retain executable candidate policies and solve a mixture linear program (LP) imposing the risk constraint for *every state*; an independent recursion checks their complete cost and risk vectors. The same locally capped menu yields mean cost 40.80 against the ordered compiler's 41.07; a wider menu with globally allocated risk reaches 40.38 (largest cell improvement 7.16%).

Matching each state's risk to the ordered incumbent, rather than allowing the entire 0.01 cap, reduces the attainable gains to 0.05% and 0.06%. Thus risk expenditure explains much of this reference improvement too. These are attainable broader-policy improvements, not upper bounds on the loss from ordering: point-based planning supplies feasible policies, not the unrestricted optimum. A two-state example in Appendix D has an ordered/general cost ratio growing as β^{1-D}. No horizon-uniform

approximation claim is made for this restriction. A separate adaptive information–replay relaxation applies to all threshold-history policies, including repeated queries. Its numerical lower bound averages 32.25, yielding an 8.81-unit upper allowance on the incumbent's gap to that broader optimum (Proposition 9). This bound is valid in real arithmetic but loose: the point-based improvements and the information lower bound leave substantial unresolved room. Neither is used to certify near-optimality outside the ordered class. Completed internal paths of the specified score tests add no information beyond their endpoint under additive expected cost; fixed-batch acceptance-test duration is constant by construction. Counts, early interruption, and other test families can carry additional information.

6.2 Compilation must earn its overhead

The three-cost frontier can grow rapidly. In the scaling measurements with $D = 5, h = 2$, compilation takes 0.36 s at $n = 16$, 2.85 s at $n = 32$, and 21.14 s at $n = 64$. These timings describe the evaluated implementation and workload. The $n = 128, D = 5$ job exceeds 90 s. At $n = 64$, only 93 records survive in the largest frontier, yet 12.3 million child pairs are generated. A completed $n = 128, D = 3, h = 8$ run solves a shallower class and matches one-step cost within numerical tolerance.

For the 64-state clustered instance with clean prior 0.8 and noise 0.25, the full and one-step model costs are 27.50 and 27.95. Incremental planning takes 21.14 seconds. Using unrounded costs, the break-even count at one millisecond per resource unit is

$$N_{\mathrm{break}} = \left\lceil \frac{T_f - T_g}{v(C_g - C_f)} \right\rceil = 47{,}022,$$

where f is the full compiler, g is one-step, and $v = 10^{-3}$ seconds per unit. This excludes unshared calibration. This planning cost is a central deployment constraint. Reusing the eight-condition mean saving would give the wrong denominator for this timing instance.

An instance-specific numerical gap can guide effort spent on finer buckets. On the same eight-condition grid, the evaluated $U - L_h$ averages 0.24 and reaches 0.47 units, against mean one-step saving 0.95 in model expectation. The largest-gap cell saves 1.70 units over one-step. Even a small certificate limits the remaining *within-class* opportunity, not the gains from better tests or broader policies. They are ordinary-double evaluations of Equation (8).

7 Executed code edits with calibrated unit-test diagnostics

We execute an authored seven-module numerical pipeline ($n = 8$) with actual Python source files, a cached assumption record, and coordinated restoration. The modules perform record-wise arithmetic; a sign mutation affects refund records. The executor uses a conservative suffix-rebuild contract after rollback, rather than arbitrary selective file repair. A valid implementation can disagree bitwise with a high-accuracy reference on cancellation inputs while satisfying its specified absolute tolerance. A brittle unit test requires exact equality, creating false-bad reports; sampling may also miss the refund mutation. Reports arise from running the code, not from flipping a hidden validity label.

A raw prefix test samples one fixture for every included module. The finite 128-row bank contains four cancellation fixtures, 32 refund fixtures, and 92 ordinary fixtures. A clean module fails exact equality on four rows and the mutant on 36. Independent sampling with replacement makes the conditional channel explicit, even though each individual test result is deterministic. This narrow, controlled mutation family lets us measure diagnostics rather than assume a semantic verifier's accuracy. Randomly generated test inputs are established testing tools (Claessen & Hughes, 2000). Floating-point behavior can also make tests unreliable (Luo et al., 2014), motivating checks of both error directions.

We collect 6,000 labeled raw checks per truth and boundary for calibration, and 2,000 new checks for held-out characterization. Labels come from exhaustive *tolerance-based* contract tests on preselected prefixes. Simultaneous confidence intervals determine safe fixed-batch acceptance services. The compiler uses conservative false-clean bounds and fitted false-bad probabilities. Its safety statement is conditional on calibration coverage and the fixture family; its cost-optimality statement applies to the fitted model. Restoration rewrites both files and assumptions and runs 256 regression cases per regenerated module.

Table 2: Executed code recovery on 800 paired setups: two clean-state priors, $D = 3$. The final two rows repair the raw test oracle using the specified tolerance, at one-percent and tighter 0.1-percent caps. These are changed checkers, included as strong engineering comparators. Retry repeats the last workflow action and leaves persistent edits intact.

Controller	Unit tests	Errors / 800
Restart	1792.00	0
Retry last action	128.00	384
One-step, symmetric	1504.36	0
One-step, asymmetric	1233.91	1
Depth-2 lookahead	1232.10	0
Full compiler	1232.10	0
Nominal compiler	1294.04	1
Tolerance oracle, 1% cap	723.05	2
Tolerance oracle, 0.1% cap	802.03	0

The experiment executes 8,000 recoveries in randomly interleaved controller order. Within the brittle-checker menu, asymmetry reduces one-step work by 17.98% (Table 2); its absolute paired reduction is 270.45 tests, interval $[248.07, 292.83]$. Full compilation's additional saving is only 1.81 tests, interval $[-4.59, 8.21]$. Depths two and three choose the same policies and execute the same actions as full compilation. The full/one-step wall-time ratio is 1.02, with a paired 20-task-block bootstrap interval $[1.01, 1.02]$: full compilation is slightly slower in this run.

The strongest comparator repairs the test oracle: replace exact floating equality by the documented tolerance. False-bad reports then disappear on the declared clean family. A separately calibrated one-step controller uses 723.05 tests and records two errors. Tightening its cap to 10^{-3} still uses only 802.03 tests, with zero observed errors and a lower true-model marginal risk than full compilation. Its 34.9% work reduction therefore persists under a stricter design constraint, rather than relying on the extra observed mistakes. Fixing a checker can save more than optimizing around its defects. The unit-test case supports this priority, while the structured score-walk case measures the smaller benefit of erroneous-branch planning when the service interface is held fixed.

Calibration costs 336,000 executed unit checks for the brittle channel, plus 112,000 held-out characterization checks. These costs are shared by same-menu planners. They must be charged in full against an uncalibrated restart policy. The tolerance-oracle alternative incurs a separate 168,000-check calibration. Cached fixtures and file setup are common pre-recovery work; random sampling, test execution, writes, and replay are included in execution timings.

8 Related work and deployment guidance

Diagnosis and search. Troubleshooting already trades observations against repair (Breese & Heckerman, 1996). Equivalence-class and decision-region methods study learning enough to act (Golovin et al., 2010; Javdani et al., 2014; Chen et al., 2015); adaptive submodularity explains important greedy guarantees (Golovin & Krause, 2011). Our edge-cutting comparison applies that approach under an explicit recovery wrapper. Optimal search trees and noisy binary search address related order and observation problems (Knuth, 1971; Chrobak et al., 2022; Ben-Or & Hassidim, 2008; Dereniowski et al., 2025). Recovery resolution and its information lower bound are developed in Appendix C, crediting classical entropy and adaptive change-of-measure arguments (Cover & Thomas, 2006; Kaufmann et al., 2016). The main novelty is the ordered erroneous-path representation and its diagnostic consequences, rather than diagnosis for action in general.

Planning and recovery infrastructure. The finite-mode compiler is a structured alternative to full belief-state value vectors (Smallwood & Sondik, 1973; Kaelbling et al., 1998), and the broader reference uses point-based planning (Pineau et al., 2003). Approximate Pareto frontiers are established (Papadimitriou & Yannakakis, 2000). Checkpoint placement, coordinated rollback, and compensating actions are separate systems concerns (Young, 1974; Daly, 2006; Elnozahy et al., 2002; Garcia-Molina & Salem, 1987; Benoit et al., 2017). AgentRewind supplies complementary recoverable execution

infrastructure (Zhuang et al., 2026). Our controller assumes that its selected restore operation is faithfully implemented.

Checker economics and reuse. Randomized verification can be cheaper than reconstruction: algebraic verification (Freivalds, 1979; Bennett et al., 2024), acceptance sampling (NIST/SEMATECH, 2012), and sampling to clean stale views (Krishnan et al., 2015) provide examples. The actual checker contract determines whether this regime applies. Rerunning a deterministic verifier on unchanged input provides no new evidence.

Start with a trusted restart and the best available test oracle. Calibrate on independently selected labeled prefixes. Compare one-step and depth-two policies at matched risk and under the relevant failure penalty. Compile only when conservative measured savings times the justified reuse count exceed incremental planning, evaluation, and calibration costs. Reprice stored policy vectors when only priors or resource prices change; recompile when their ranking or safety-feasible menu changes. For fixed-policy saving vector g, prior shift measured by total variation (TV), $\mathrm{TV}(b, b') \le e$, gives $b'^T g \ge b^T g - e(\max g - \min g)$, a cheap reuse check. Channel drift requires a new valid error bound, stronger tests, or trusted restart. A new prior alone cannot invalidate statewise safety.

9 Conclusion

Asymmetric diagnostic design gives the largest measured improvement for checkpointed recovery. Three-cost compilation correctly accounts for mistaken branches, extends to failure penalties and finite persistent modes, and gives a useful restricted-class reference. Stronger inexpensive baselines, risk matching, and real code execution narrow its practical advantage: improve the test oracle first, use shallow lookahead next, and reserve compilation for workflows where its remaining benefit can repay its overhead.

AI USE STATEMENT

In this work, we used generative AI tools to draft sections of the paper and polish writing. Additionally, we used generative AI tools for editing code. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Michael Ben-Or and Avinatan Hassidim. The Bayesian learner is optimal for noisy binary search (and pretty good for quantum as well). In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science*, pp. 221–230, 2008. doi: 10.1109/FOCS.2008.58.

Huck Bennett, Karthik Gajulapalli, Alexander Golovnev, and Evelyn Warton. Matrix multiplication verification using coding theory. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, volume 317 of *Leibniz International Proceedings in Informatics*, pp. 42:1–42:13, 2024. doi: 10.4230/LIPIcs.APPROX/RANDOM.2024.42.

Anne Benoit, Saurabh K. Raina, and Yves Robert. Efficient checkpoint/verification patterns. *The International Journal of High Performance Computing Applications*, 31(1):52–65, 2017. doi: 10.1177/1094342015594531.

John S. Breese and David Heckerman. Decision-theoretic troubleshooting: A framework for repair and experiment. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pp. 124–132, 1996. URL https://arxiv.org/abs/1302.3563.

Yuxin Chen, Shervin Javdani, Amin Karbasi, J. Andrew Bagnell, Siddhartha Srinivasa, and Andreas Krause. Submodular surrogates for value of information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015. doi: 10.1609/aaai.v29i1.9694.

Marek Chrobak, Mordecai Golin, J. Ian Munro, and Neal E. Young. A simple algorithm for optimal search trees with two-way comparisons. *ACM Transactions on Algorithms*, 18(1):1–11, 2022. doi: 10.1145/3477910.

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*, pp. 268–279, 2000. doi: 10.1145/351240.351266.

C. J. Clopper and E. S. Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4):404–413, 1934. doi: 10.1093/biomet/26.4.404.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, second edition, 2006. doi: 10.1002/047174882X.

J. T. Daly. A higher order estimate of the optimum checkpoint interval for restart dumps. *Future Generation Computer Systems*, 22(3):303–312, 2006. doi: 10.1016/j.future.2004.11.016.

Dariusz Dereniowski, Aleksander Łukasiewicz, and Przemysław Uznański. Noisy (binary) searching: Simple, fast and correct. In *42nd International Symposium on Theoretical Aspects of Computer Science*, volume 327 of *Leibniz International Proceedings in Informatics*, pp. 29:1–29:18, 2025. doi: 10.4230/LIPIcs.STACS.2025.29.

E. N. (Mootaz) Elnozahy, Lorenzo Alvisi, Yi-Min Wang, and David B. Johnson. A survey of rollback-recovery protocols in message-passing systems. *ACM Computing Surveys*, 34(3):375–408, 2002. doi: 10.1145/568522.568525.

Rūsiņš Freivalds. Fast probabilistic algorithms. In *Mathematical Foundations of Computer Science*, pp. 57–69. Springer, 1979. doi: 10.1007/3-540-09526-8_5.

Hector Garcia-Molina and Kenneth Salem. Sagas. In *Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data*, pp. 249–259, 1987. doi: 10.1145/38713.38742.

Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42:427–486, 2011. doi: 10.1613/jair.3278.

Daniel Golovin, Andreas Krause, and Debajyoti Ray. Near-optimal Bayesian active learning with noisy observations. In *Advances in Neural Information Processing Systems*, volume 23, 2010. URL https://papers.nips.cc/paper_files/paper/2010/hash/1e6e0a04d20f50967c64dac2d639a577-Abstract.html.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. doi: 10.1080/01621459.1963.10500830.

Shervin Javdani, Yuxin Chen, Amin Karbasi, Andreas Krause, Drew Bagnell, and Siddhartha Srinivasa. Near optimal Bayesian active learning for decision making. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, volume 33 of *Proceedings of Machine Learning Research*, pp. 430–438, 2014. URL https://proceedings.mlr.press/v33/javdani14.html.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998. doi: 10.1016/S0004-3702(98)00023-X.

Emilie Kaufmann, Olivier Cappé, and Aurélien Garivier. On the complexity of best-arm identification in multi-armed bandit models. *Journal of Machine Learning Research*, 17(1):1–42, 2016. URL https://www.jmlr.org/papers/v17/kaufman16a.html.

Donald E. Knuth. Optimum binary search trees. *Acta Informatica*, 1:14–25, 1971. doi: 10.1007/BF00264289.

Sanjay Krishnan, Jiannan Wang, Michael J. Franklin, Ken Goldberg, and Tim Kraska. Stale view cleaning: Getting fresh answers from stale materialized views. *Proceedings of the VLDB Endowment*, 8(12):1370–1381, 2015. doi: 10.14778/2824032.2824037.

Qingzhou Luo, Farah Hariri, Lamyaa Eloussi, and Darko Marinov. An empirical analysis of flaky tests. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp. 643–653, 2014. doi: 10.1145/2635868.2635920.

NIST/SEMATECH. Choosing a sampling plan with a given OC curve. e-Handbook of Statistical Methods, Section 6.2.2.2, 2012. URL https://www.itl.nist.gov/div898/handbook/pmc/section2/pmc222.htm. Handbook revision; accessed September 25, 2026.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023. URL https://arxiv.org/abs/2310.08560.

Christos H. Papadimitriou and Mihalis Yannakakis. On the approximability of trade-offs and optimal access of Web sources. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pp. 86–92, 2000. doi: 10.1109/SFCS.2000.892068.

Joelle Pineau, Geoffrey Gordon, and Sebastian Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pp. 1025–1032, 2003. URL https://publications.ri.cmu.edu/point-based-value-iteration-an-anytime-algorithm-for-pomdps.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Richard D. Smallwood and Edward J. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21(5):1071–1088, 1973. doi: 10.1287/opre.21.5.1071.

Abraham Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16 (2):117–186, 1945. doi: 10.1214/aoms/1177731118.

Abraham Wald and Jacob Wolfowitz. Optimum character of the sequential probability ratio test. *The Annals of Mathematical Statistics*, 19(3):326–339, 1948. doi: 10.1214/aoms/1177730197.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629.

John W. Young. A first order approximation to the optimum checkpoint interval. *Communications of the ACM*, 17(9):530–531, 1974. doi: 10.1145/361147.361115.

Yu Zhuang, Kefei Chen, Yitong Duan, Shuxin Zheng, Jian Li, and Xu-Yao Zhang. AgentRewind: Recoverable execution for long-horizon LLM agents. arXiv:2608.14380, 2026. URL https://arxiv.org/abs/2608.14380.

A DIAGNOSTIC SAFETY, CALIBRATION, AND PERSISTENT MODES

A.1 INDEXING AND COMPLETE SERVICE SEMANTICS

Fault states in the paper are $1, \dots, n$ and checkpoint actions are $0, \dots, n-1$. Under zero-based fault-state and boundary indexing, the same unsafe event occurs when the checkpoint action exceeds the fault-state index. A service at boundary a terminates almost surely and has finite conditional mean cost. Its expected duration can depend on prefix truth and be correlated with its final report. For the three-cost theorem, continuation depends on the report only; future service randomness is conditionally fresh. An invocation begins at recovery and ends at its terminal rollback. Multiple invocations require a separate lifetime risk allocation.

A.2 PROOF OF DIRECTIONAL SAFETY

Proof of Theorem 1. Fix state j. Before the first erroneous report, the retained interval contains j. At a bad threshold, $j \le a$, a false-clean report sends the controller to an interval whose left endpoint exceeds a. Every descendant checkpoint is at least $a \ge j$ and hence unsafe. At a good threshold, $j > a$, a false-bad report sends the controller to an interval whose right endpoint is at most a. Every descendant checkpoint is at most $a - 1 < j$, hence safe. If no error occurs, the left endpoint minus one is safe. These three disjoint events partition the outcome.

For a fixed truthful path, the probability that its first error occurs at a particular node is the product of preceding correct-report probabilities times that node's error probability. Only first false-clean errors count toward failure, giving Equation (4). When service errors depend on history, replace the product by the actual probability of reaching the node without a previous error. Conditional on such a history, its false-clean probability is bounded by $\bar{\alpha}_v$. Summing these disjoint first-error probabilities and bounding each reach probability by one gives the stated sum. Independence is thus sufficient for the product formula, but conditional history-uniform error bounds suffice for the inequality. A marginal bound with no conditioning guarantee does not suffice. □

The proof relies on the conservative descendant action set. With arbitrary terminal actions or backtracking, an initial conservative report can be followed by an unsafe action, so that argument must be replaced by full statewise evaluation. Equation (4) also explains why equal δ/D allocation can be wasteful: some dangerous nodes are rarely reached or do not lie on a state's truthful path.

A.3 STOPPED SCORE TESTS AND THE MEANING OF FRESH EVIDENCE

Let $S_0 = 0$ and $S_t - S_{t-1} \in \{-1, +1\}$. The probability of an upward step is $1 - \eta$ under bad truth and η under good truth. Stop on first reaching $-l$ or $+u$. A block of $l + u$ upward steps causes absorption from every transient state and has positive probability under either truth. Therefore the stopping time has a geometric block tail, and its mean is finite. Under bad truth, $z^{S_{t\wedge\tau}}$, with $z = \eta/(1-\eta)$, is a bounded martingale. Optional stopping gives

$$1 = \alpha z^{-l} + (1-\alpha) z^{u},$$

which yields α in Equation (5). Reflecting the walk gives β. The mean score drift gives

$$(1-2\eta)\mathbb{E}_b\tau = u - (l+u)\alpha, \qquad (1-2\eta)\mathbb{E}_g\tau = l - (l+u)\beta.$$

This proves all four formulas. If $\eta = 0$, one perfect raw observation suffices. Test costs multiply these counts by the query price; the controller is never charged only one unit for an entire sequential test. These are classical sequential-testing calculations (Wald, 1945; Wald & Wolfowitz, 1948).

Proposition 5 (Completed transcript sufficiency)**.** *For this binary-symmetric score walk, conditional on its final endpoint, the entire stopped path and its duration have the same distribution under bad and good truth. For additive expected cost, fresh future services, no deadline, and randomized continuation policies, using the completed internal path provides no additional decision information beyond the report.*

Proof. A path with x upward and y downward steps has likelihood ratio $((1-\eta)/\eta)^{x-y}$ between bad and good truth. At absorption $x - y$ is the reported endpoint, either u or $-l$. The ratio is constant

over paths with the same endpoint. Normalizing within the endpoint event gives equal conditional path laws. A report-only policy can sample a synthetic path from this common law and apply the original continuation. For each state, this reproduces the distribution of future actions. Past costs are sunk and additive; their marginal means are unchanged. Repeating this construction for each completed service gives equal expected cost and terminal risk. A deadline, early service interruption, or duration-dependent future channel breaks this equivalence. □

The code checker uses a *fixed* batch size r, so its raw-query duration is constant by construction. Its count of failing checks can still carry information beyond its binary acceptance report. That count is excluded from the basic compiler, while the broader-history reference also receives completed binary reports. Proposition 5 is not used to discard count information for fixed-batch tests.

A.4 Calibration and finite acceptance services

Let $Y = 1$ denote a failing raw check. On a fixed labeled prefix family suppose $\mathbb{P}(Y = 1 \mid \mathrm{bad}) = p_b$ and $\mathbb{P}(Y = 1 \mid \mathrm{good}) = p_g$. A fixed-size test draws r fresh checks and reports bad when their count exceeds c. Its errors and deterministic number of raw checks are

$$\alpha = \mathbb{P}\{\mathrm{Bin}(r, p_b) \le c\}, \qquad \beta = \mathbb{P}\{\mathrm{Bin}(r, p_g) > c\}, \qquad N = r. \tag{9}$$

This is an operating-characteristic calculation from acceptance sampling (NIST/SEMATECH, 2012). Use simultaneous confidence intervals $[L_b, U_b]$, $[L_g, U_g]$ from independently sampled reference checks. Because the binomial count increases stochastically with its success probability, substituting L_b upper-bounds false-clean risk and substituting U_g upper-bounds false-bad probability. Search a predeclared finite set of (r, c) using these same intervals. Coverage is simultaneous over all selected designs, so data-dependent menu construction does not require an extra union bound over (r, c).

Corollary 6 (Calibrated statewise safety)**.** *Suppose the reference checks have the deployment channel law, and all interval bounds jointly cover with probability at least $1 - \xi$. Select only services whose upper false-clean bound is at most δ/D, or use trusted restart when none exists. Conditional on simultaneous coverage, every selected ordered tree has statewise unsafe probability at most δ. Averaging over calibration, risk is at most $(1 - \xi)\delta + \xi$. Fitted false-bad probabilities and costs need not be exact for this safety claim; the cost-optimality theorem remains a statement about the supplied model.*

Proof. On the coverage event, monotonicity of the binomial tail makes every selected service's true false-clean probability at most its reported upper bound. Apply Theorem 1 simultaneously to all menu trees. Restart is safe regardless of coverage. Bounding risk by one off the coverage event gives the unconditional expression. □

The code experiments use Clopper–Pearson bounds (Clopper & Pearson, 1934), not the model's hidden probabilities, to construct the menu. Known fixture masses are used only for separate analytic evaluation. A Hoeffding upper bound (Hoeffding, 1963) would provide another finite-sample construction. Neither method turns selectively discovered production failures into an unbiased calibration sample. Representative checks must be selected independently of the deployed recovery controller, and a reliable labeling procedure must be available. The fixed finite fixture family makes that source of truth explicit here.

A.5 Correlated errors: a bound and a computable model

Proof of Proposition 2. For each state, let $r_j^0 \le \delta_0$ be fresh risk and $r_j^Q \le 1$ contamination risk. Equation (6) gives

$$r_j = (1 - \gamma_j) r_j^0 + \gamma_j r_j^Q \le (1 - \gamma_j)\delta_0 + \gamma_j \le (1 - \bar{\gamma})\delta_0 + \bar{\gamma}.$$

A two-state tree with fresh failure probability δ_0 and a contaminated channel that always selects its unsafe leaf attains equality. Mixing an independent safe restart with probability q multiplies the bound by $1 - q$. Solving either inequality for δ_0 or q gives the stated rules. The argument permits any correlation in Q_j; no raw-query independence is invoked inside that mode. □

A bound on $\bar{\gamma}$ can come from a justified failure-mode model or separately labeled episode-level checks. Repeated observations of one apparently correct prefix do not estimate the probability of a hidden common-mode blind spot. If the calibration coverage failure is also possible, it must be added as a separate bad event. Similarly, an independently justified replay-failure probability ρ_r gives the conservative union bound $r_j \le r_j^{\text{diagnostic}} + \rho_r$; the diagnostic theorem alone covers only faithful regeneration.

Proof of Corollary 4. Condition on a fixed mode m. At an interval I, every state inside I gives the same truth to each ancestor comparison. Conditional on the mode, the likelihood of the ancestor reports is therefore constant over those inside states. Their distribution is $b^{(m)}$ restricted to I. All states to the left give bad truth to every internal comparison, and all states to the right give good truth. Thus the same three-cost recurrence applies mode by mode, with its own prior split p_m and its own service parameters.

At each candidate split, the same two executable subtrees and service identity are used for every mode. Concatenating the mode triples retains their different costs while enforcing this shared policy. A componentwise cheaper $3m$-vector cannot make any parent or any nonnegative mode-weighted root objective larger. Exhaustive enumeration and dominance pruning therefore preserve an optimum. At the root, the distribution over modes is its original prior, so the objective is $\sum_m \mathbb{P}(M = m)A_m$. The optimizer never reads the realized mode. Safety can be imposed by modewise false-clean bounds, the mixture bound, or full statewise evaluation; cost dominance alone does not enforce arbitrary additional risk constraints. □

The implementation validates two-mode costs against explicit enumeration of all ordered trees on 12 four-state cases. A further 18 settings at four/eight states compare independent-channel planning with exact persistent-mode planning. The largest penalized-objective improvement is only 0.04 units; the theoretical extension is not presented as a large empirical effect. The unknown-mode law permits dependence across services through the single latent variable; conditional freshness within each mode remains part of the model.

B THREE-COST COMPILATION: FULL PROOF AND COMPLEXITY

B.1 EXACT REPRESENTATION AND FAILURE PENALTIES

Proof of the exact part of Theorem 3. For an interval $[i, j]$, the inside prior is b restricted to $[i, j]$. The inside state is in the left child with probability p. Under that event the service has mean cost t^b. A correct bad report enters the left child's inside population, while a false-clean report enters the right child's earlier-state population. The conditional expected remaining objective is consequently $t^b + (1 - \alpha)A_L + \alpha B_R$. For an inside state in the right child, the corresponding expression is $t^g + \beta C_L + (1 - \beta)A_R$. Averaging these expressions gives the first line of Equation (7).

A state earlier than the parent interval makes the selected comparison bad and is earlier than either child, giving its B recurrence. A state later than the parent makes the comparison good and remains later than either child, giving its C recurrence. The correlation between service cost and report does not alter these sums: linearity charges the unconditional conditional-on-truth mean cost once, and future continuation depends only on the report. At a leaf $i - 1$, earlier states incur $R_{i-1} + \lambda$ and all other represented states incur R_{i-1}. This establishes the base vector and induction for the penalized objective.

A vector dominated in all coordinates is replaceable by its dominating executable subtree in every possible parent, since all recurrence coefficients are nonnegative. Induct on remaining depth. Every tree is either a leaf or a split/service with two smaller subtrees. Enumeration considers all such choices; pruning preserves an equally good or better candidate. At the root, there are no outside states, so minimizing A minimizes the full objective in $\mathcal{T}_D(\mathcal{S})$. Every generated tree has the local safety cap and Theorem 1 applies independently of its cost or penalty.

If an interval has zero prior mass, its inside coordinate has zero weight at its first ancestor containing positive mass. Define its artificial inside distribution uniformly over its states and use the induced splits consistently. Outside coordinates remain exact. This convention does not change the root value

and avoids division by zero. The implementation clamps numerically negative prefix-sum roundoff to zero. □

The nominal comparator removes erroneous-branch transitions while retaining the exact same menu and truth-conditional service costs. If $w_{ij} = \sum_{k=i}^{j} b_k$, its unnormalized recurrence is

$$V(i,j,d) = \min\left\{ w_{ij} R_{i-1},\ \min_{a,s}\{ w_{ia} t_s^b + w_{a+1,j} t_s^g + V(i,a,d-1) + V(a+1,j,d-1) \} \right\}.$$

This is a truthful-path ablation, not a recommended noisy controller. All final comparator trees are evaluated and executed with the actual observation law.

B.2 Feasible bucketing and the optimistic lower recurrence

At one subproblem, assign a candidate (A, B, C) to the fixed bucket $(\lfloor B/h \rfloor, \lfloor C/h \rfloor)$. Keep the candidate with smallest A and its executable tree. A replacement within a bucket cannot increase A, and can increase either tail by less than h.

Proof of the additive and bracket parts of Theorem 3. Inductively suppose every exact child vector is dominated from above, within additive $(d-1)h$ per coordinate, by some retained feasible child vector. Nonnegative coefficients whose child weights sum to one imply that combining these representatives increases every parent coordinate by at most $(d-1)h$. Bucketing adds at most h to the tails and nothing to the inside coordinate. The claim follows at depth d. A dominated candidate can again be replaced by a cheaper survivor. Leaves have zero error. At the root, the selected inside objective is at most Dh above optimum, and feasibility makes it at least the optimum. No service probability or risk is rounded.

For the lower bound, instead store the minimum candidate A in each bucket together with that bucket's two lower edges. These triples may be non-executable. Leaves start exact. Suppose an exact child has some stored triple no larger in every coordinate. Combining such triples under Equation (7) gives a candidate no larger than the exact parent. Replacing it by minimum A and lower tail edges preserves this property; dominance pruning also preserves it. At the root some retained lower triple lies below an exact optimum, hence the minimum lower A is a lower bound. Evaluating any feasible tree gives the upper bound U. □

For $\lambda > 0$, all coordinates are at most $B_D^\lambda = R_0 + \lambda + D \max_s \max(t_s^b, t_s^g)$, so the same proof and bucket analysis apply. This bounds expected objective coordinates, not the duration of an individual sequential test.

Numerical meaning. Both statements above are real-arithmetic results. The implementation uses double-precision arithmetic, floating prefix sums, and numerical linear programming. It checks inequalities to documented tolerances and compares small instances against enumeration. It does not implement interval arithmetic, directed rounding, or a proof assistant. The reported $U - L_h$ is an informative numerical evaluation of a proved bound, not a machine-verified enclosure. A small gap limits additional optimization opportunity within this class; it cannot bound the gain from a better checker or a less restricted controller.

Storage and candidate generation. There are $O(Dn^2)$ interval/depth subproblems. Each has at most $K_h = (\lfloor B_D^\lambda / h \rfloor + 1)^2$ retained bucket entries. With at most m services per boundary, a subproblem can generate $1 + O(nmK_h^2)$ child-pair candidates. This roughly quartic dependence on B_D^λ / h can be prohibitive. The implementation streams fixed-bucket candidates in blocks of at most 4,096 pairs instead of materializing every child Cartesian product. This controls temporary arrays, not the total number of pairs. Exact unbucketed frontiers have no stated polynomial cardinality bound. The $3m$ persistent-mode reference uses exact dominance without asserting comparable scaling.

Incumbents and computational abstention. A cheaper evaluated one-step incumbent can replace the bucketed tree without weakening the additive guarantee. The comparison concerns model cost; Monte Carlo execution can reverse a tiny difference. A timeout returns no approximate optimum or fabricated timing. Deployment should retain its already available baseline rather than wait indefinitely

for compilation. An optional lower-certificate pass costs additional computation and is charged separately in amortization.

C Information needed for recovery, including a broad-class bound

C.1 Exact noiseless recovery resolution

For an excess allowance $\epsilon \geq 0$, define

$$\mathcal{A}_j^\epsilon = \{k < j : R_k - R_{j-1} \leq \epsilon\}.$$

These are the safe actions with small excess work. Let M_ϵ be the minimum number of consecutive intervals covering the state set such that each interval $[i, j]$ has $R_{i-1} - R_{j-1} \leq \epsilon$.

Theorem 7 (Noiseless recovery resolution)**.** *M_ϵ equals the largest number of states whose oracle replay costs are pairwise more than ϵ apart. Extending each consecutive interval as far as allowed computes this number. The minimum worst-case number of noiseless threshold queries for deterministic safe ϵ-excess recovery is $\lceil \log_2 M_\epsilon \rceil$.*

Proof. Let greedy interval starts be $s_1, \ldots, s_M$. Maximal extension means $R_{s_r-1} - R_{s_{r+1}-1} > \epsilon$. They form a separated packing by monotonicity. No allowed interval contains two packed states, while no packing contains two points from a greedy interval. Thus packing, greedy, and minimum cover counts coincide. If $e < \ell$ have cost gap greater than ϵ, an action safe for e costs too much for ℓ, so their acceptable action sets are disjoint. A depth-q binary decision tree has at most 2^q leaves, yielding $q \geq \lceil \log_2 M \rceil$. Conversely, binary search over the ordered greedy groups has that depth, and action $i - 1$ is safe with the allowed excess for group $[i, j]$. □

For oracle costs $(101, 100, 2, 1)$ and $\epsilon = 1$, only two recovery groups are needed although four exact fault states remain. The result specializes the distinction between identification and decision-making; general decision-region methods treat richer overlap patterns (Javdani et al., 2014; Chen et al., 2015).

C.2 Adaptive information–recomputation lower bound

Let KL denote Kullback–Leibler divergence. Assume independent raw binary-symmetric threshold channels with $0 < \eta_a < 1/2$, raw prices c_a, and finite expected total observations. Define

$$d_a = (1 - 2\eta_a) \log \frac{1 - \eta_a}{\eta_a}, \qquad U_\ell = \mathbb{E}_\ell (R_K - R_{\ell-1})_+.$$

For $0 < \delta < 1/2$, let $\phi_\delta(q) = \mathrm{kl}(q, \delta)$ for $q > \delta$ and zero otherwise. Write $\Delta_{e\ell} = R_{e-1} - R_{\ell-1} > 0$.

Theorem 8 (Information or conservative replay)**.** *Every uniformly δ-safe adaptive policy whose informative observations are these channels satisfies, for each $e < \ell$ with positive gap,*

$$\sum_{a=e}^{\ell-1} d_a \mathbb{E}_\ell N_a \geq \phi_\delta\big([1 - U_\ell/\Delta_{e\ell}]_+\big)\,. \tag{10}$$

Queries can be reordered, repeated, or interrupted. No ordered-tree restriction is used in this bound.

Proof. Use the complete stopped transcript including randomized controls and the final action. Given the same history, controller action kernels are identical in worlds e and ℓ. Thresholds outside $[e, \ell - 1]$ have identical observation laws; separating thresholds have divergence d_a. The stopped chain rule yields $\mathrm{KL}(P_\ell^{\mathrm{tr}} \| P_e^{\mathrm{tr}}) = \sum_{a=e}^{\ell-1} d_a \mathbb{E}_\ell N_a$. This follows by truncation with an absorbing stop symbol, the finite chain rule, and convergence of the nonnegative expected information sum. It is the standard adaptive change-of-measure technique (Kaufmann et al., 2016).

For event $A = \{K \geq e\}$, safety gives $P_e(A) \leq \delta$. On its complement in state ℓ, replay excess is at least $\Delta_{e\ell}$, so $P_\ell(A) \geq 1 - U_\ell/\Delta_{e\ell}$. Data processing into $\mathbf{1}_A$ gives the binary KL lower bound. For $q > \delta$, the minimum of $\mathrm{kl}(u, v)$ over $u \geq q, v \leq \delta$ is $\mathrm{kl}(q, \delta)$; for $q \leq \delta$ use zero. This proves the claim. □

Full restart pays zero information and large excess work, an essential feasible endpoint. A separate free verifier or informative replay would have to enter the transcript calculation; the bound is not applied while ignoring such an information source.

Proposition 9 (Finite LP lower bound and a broad-class gap allowance)**.** *For each state ℓ, choose finitely many supporting tangents to the convex functions $h_{e\ell}(u) = \phi_\delta([1 - u/\Delta_{e\ell}]_+)$ on $u \geq 0$. Minimize $\sum_{a<\ell} c_a x_a + u$ over $x_a \geq 0$, $0 \leq u \leq R_0 - R_{\ell-1}$ subject to*

$$\sum_{a=e}^{\ell-1} d_a x_a \geq h_{e\ell}(u_0) + h'_{e\ell}(u_0)(u - u_0)$$

for all chosen tangents. Add $R_{\ell-1} - \delta(R_{\ell-1} - R_{n-1})$ and call the result L_ℓ. Then

$$L_{\mathrm{all}} = \sum_{\ell} b_\ell L_\ell \leq C^*_{\mathrm{all}}, \qquad 0 \leq C(\widehat{\pi}) - C^*_{\mathrm{all}} \leq C(\widehat{\pi}) - L_{\mathrm{all}}, \tag{11}$$

for any feasible incumbent $\widehat{\pi}$. C^*_{all} permits all adaptive threshold policies under the same expected-work objective and uniform risk cap.

Proof. The binary KL function is convex; composition with $1 - u/\Delta$ and continuation as zero at $q \leq \delta$ preserves convexity. Each tangent is a lower bound. The actual vector $(\mathbb{E}_\ell N_a, U_\ell)$ satisfies Equation (10), hence every tangent constraint. Ignoring queries outside the separating region can only reduce the objective. Also

$$\mathbb{E}_\ell R_K = R_{\ell-1} + U_\ell - \mathbb{E}_\ell(R_{\ell-1} - R_K)_+ \geq R_{\ell-1} + U_\ell - \delta(R_{\ell-1} - R_{n-1}),$$

because cheaper-than-oracle replay requires an unsafe action. Each state's actual expected cost thus exceeds its LP value plus the stated base term. Averaging proves the first bound; subtracting it from any feasible cost proves the second. □

We use 33 evenly spaced tangent probabilities between $\delta + 10^{-4}$ and $1 - 10^{-4}$, supplemented by $1 - 10^{-2}$, $1 - 10^{-3}$, and $1 - 10^{-4}$. The resulting bound is deliberately a relaxation: it need not describe count vectors achievable by one controller. At $\delta = 0.01$, its grid mean is 32.25 against the ordered incumbent's 41.07, leaving a valid but loose 8.81-unit mean gap allowance. This does not certify that the observed point-based reference has found the unrestricted optimum. Eight small cases compare the bound with exact arbitrary-history LPs; 24 grid cases compare it with feasible ordered policies. Values are numerical LP evaluations rather than machine-certified bounds.

A necessary risk domain for two-point statements. If two states have disjoint acceptable action sets and each must have success probability at least $1 - \delta$, the standard two-point KL bound uses $0 < \delta < 1/2$. At $\delta = 1/2$, a zero-query random choice between their respective actions can meet that success target. We retain the strict restriction wherever invoking this separation. Binary entropy/packing extensions follow the usual information inequalities (Cover & Thomas, 2006); the reported experiments do not rely on an omitted $\delta \geq 1/2$ extension.

D Baseline algorithms and broader recovery classes

D.1 Receding lookahead with actual report probabilities

The full posterior has weights for every fault state, including states outside the nominal interval after a mistaken report. At each node, a depth-h lookahead enumerates all legal split/service choices for h further decisions. A truncated leaf uses the conservative interval action. It evaluates its cost and risk penalty exactly under those state weights, implements the first chosen action, then repeats with remaining horizon and updated weights. Thus depth one equals the provided one-step baseline; $h \geq D$ equals exact unbucketed ordered compilation. Depth two or three is not merely a short fixed tree executed once. Its construction includes every reachable replanning node. The implementation tests these limiting equivalences across randomized small cases and multiple failure penalties. Receding-horizon values need not improve monotonically with h in every individual instance.

D.2 AN EXPLICIT EDGE-CUTTING ADAPTATION

We implement the cross-class edge objective of EC^2 (Golovin et al., 2010). A sampled hypothesis consists of a fault state plus a pre-sampled outcome for every boundary/service action. An ordered path uses each action at most once, so this finite noise-realization table suffices. State/noise hypotheses are sampled from the specified prior and channel, independently of evaluation trajectories. Hypotheses are grouped by an ϵ-recovery partition. Only pairs in different groups receive edges, weighted by the product of their empirical probabilities. A test eliminates an edge when either endpoint is inconsistent with its report. The greedy score is expected eliminated edge mass divided by expected paid service cost.

For a version space with class counts n_k and total N, the number of unresolved cross-class edges is $(N^2 - \sum_k n_k^2)/2$. We average remaining edge counts under the predicted report distribution to compute the gain. We use 2,048 particles and choose $\epsilon \in \{0, 5, 10, 20, 40\}$ by true-model expected recovery cost. That favorable selection gives the baseline a fixed declared hyperparameter search; its complete search time is charged. It stops when one recovery class remains, the depth is exhausted, or the interval is a singleton. Conservative interval leaves and locally safe services provide its risk control. The identification approximation theorem applies to the original objective and assumptions; we claim no inherited approximation factor for the recovery wrapper. The baseline is more informative than describing generic greedy information gain as EC^2.

The information-gain comparator maximizes the mutual information between fault state and binary report per expected test cost. It uses the same interval and safety wrapper. It is inspired by Bayesian noisy search (Ben-Or & Hassidim, 2008; Dereniowski et al., 2025), rather than a reproduction of either paper's optimal query backend. The experimental conclusions are explicitly about these implemented policies, not every algorithm in those broader families.

D.3 POINT-BASED REFERENCE AT SIXTEEN STATES

The reference implements finite-horizon point-based value iteration (Pineau et al., 2003) for static hidden fault states. It has all terminal checkpoint actions, all completed services at every history, and no ordering constraint. Each alpha-vector represents an executable policy. We choose 256 beliefs from the prior, state vertices, and simulated reachable posteriors, using two fixed seeds. For penalties $50, 200, 1000, 5000, 20000$, depth-five backups choose an existing continuation alpha-vector for each observation. Returned statewise cost and risk vectors are reconstructed by a separate evaluator rather than inferred from the training beliefs.

The candidate pool includes restart and the ordered baselines. For weights w_t on this pool, solve

$$\min_{w \geq 0} \sum_t w_t b^T c_t, \qquad \sum_t w_t = 1, \qquad \sum_t w_t r_t(j) \leq \delta \quad \text{for all } j. \tag{12}$$

This is a feasible randomized policy with statewise safety in the specified model. It supplies an upper bound on the broader optimum and a lower bound on an *attainable* improvement over the ordered incumbent. It does not supply a worst-case upper bound on the ordered restriction. Equation (11) provides a separate, looser upper allowance.

The two menu conditions retain either the original local δ/D cap or an extra service at each boundary that only meets $\alpha \leq \delta$. The pool LP allocates risk globally in both cases. Results average eight equal-weight configurations, using the ratio of unrounded mean costs for percentages. The same-menu improvement is 0.64%; wider-menu improvement is 1.67%, with maximum cell improvement 7.16%. A second pool LP replaces the scalar cap by the ordered incumbent's entire statewise risk vector. The same/wider menus then give costs 41.05 and 41.04, gains 0.05% and 0.06%, respectively; only one of eight cases improves. This controls risk expenditure even more strictly than a matched marginal target, while remaining a finite-library reference. Each retained mixture is independently checked against every statewise risk constraint.

D.4 AN EXACT SMALL-HISTORY OBJECTIVE AND ITS CONSTRAINTS

For completeness, let $L_j(h)$ be the observation likelihood of a finite completed history h under state j, excluding controller randomization. Realization weight $x(h, a)$ is the product of controller choice

probabilities along h and at its next action. Root flow is one. For every test action a and each possible report y, flow obeys

$$\sum_u x(hay, u) = x(h, a), \qquad x(h, a) \geq 0. \tag{13}$$

The observation probability belongs in $L_j(hay)$, not in this equality. With at most D services and compulsory terminal recovery, the exact expected-cost program is

$$\min_x \quad \sum_{h,a,j} b_j L_j(h) x(h, a) c_j(a), \tag{14}$$

$$\text{subject to} \quad \sum_{h,\, a=\text{rollback}(k)} L_j(h) x(h, a) \mathbf{1}\{k \geq j\} \leq \delta \quad (1 \leq j \leq n), \tag{15}$$

with Equation (13) and root flow. Test actions cost their conditional mean and rollback costs R_k.

Proposition 10 (Finite-history equivalence)**.** *Equations* (14)–(15)*, together with the flow equalities and root normalization, give the exact optimum over randomized policies in this finite history class.*

Proof. Every behavioral policy gives nonnegative realization weights satisfying flow. Conversely, normalize outgoing weights at each positive-flow history to obtain a behavioral policy; zero-flow choices are arbitrary. Induction on history length recovers the same weights. State-j probability of a history/action is $L_j(h)x(h, a)$. Summing paid costs and unsafe terminal probabilities therefore yields precisely the objective and constraints. This establishes both directions. The cap is on services, each of which can have unbounded internal duration but finite mean. The independent evaluator checks terminal probability one and every statewise risk. □

The small-history LP is exponential and serves as a correctness reference. It addresses expected cost, unlike a deadline-completion objective.

D.5 Why ordering can be arbitrarily expensive

Take two states with replay costs $R_0 = R, R_1 = 0$, bad-state prior probability $\pi > 0$, and a service with cost c, $\alpha = 0$, and $0 < \beta < 1$. An ordered tree can inspect at most once, so its optimum is

$$\min\{R,\ c + [\pi + (1 - \pi)\beta]R\}.$$

A general policy repeats until a clean report or D bad reports, then uses the only safe action consistent with this rule. It is perfectly safe, with cost

$$c\left[\pi D + (1 - \pi)\frac{1 - \beta^D}{1 - \beta}\right] + [\pi + (1 - \pi)\beta^D]R.$$

The formulas sum geometric reach probabilities. Setting $\pi = \beta^{2D}$ and $c/R = \beta^{2D}/D$ makes the ordered/general ratio grow on the order of β^{1-D}. Thus the restricted-class compiler theorem gives no horizon-uniform approximation ratio to all safe policies, even with perfect false-clean control.

E Code-edit workflow and the source of labels

The executed workflow comprises seven stages: amount parsing, unit conversion, discounts, fees, taxes, totals, and reporting. Each applies record-wise arithmetic with a stage-specific scale. Module bodies are small on purpose: the study measures diagnostics and recovery, not program synthesis. A mutation reverses a sign on refund records. The controller receives no fault-location metadata.

Fixture construction and truth. Each generated bank has 128 records. Four have cancellation-sensitive values, including a large positive quantity, its negation, and a small residual. Their reassociated floating computation differs bitwise from `math.fsum` but meets the declared stage-scaled absolute tolerance. Thirty-two refund rows distinguish the sign mutation, and 92 ordinary exactly representable rows pass both versions. The task seed varies the actual values within these designed categories. The generated source is compiled and executed for each task. Prefix truth is established by the full finite-domain tolerance suite, not by the brittle checker or the corrupted cache.

For boundary a in zero-based code indexing, independent per-module samples give

$$p_g(a) = 1 - (124/128)^{a+1}, \qquad p_b(a) = 1 - (92/128)(124/128)^a.$$

The bad-module factor is constant across its possible locations within the prefix. This deliberately controlled homogeneity makes the truth-dependent service model exact on this family. It is not a general property of arbitrary repository test suites.

Calibration and held-out checks. For seven boundaries and two truths, calibration executes 6,000 raw prefix reports per pair. Their total 84,000 prefix reports perform 336,000 module tests. Truth labels are assigned before query sampling by complete trusted contract checks. Confidence family size is fourteen and coverage error is 0.01. A disjoint held-out stream runs 2,000 reports for each pair (28,000 prefix reports, 112,000 module tests). Held-out counts characterize the channel and are not used to choose services. True probabilities above are used for analytic evaluation, not menu construction.

The menu searches $r \in \{8, 16, \ldots, 768\}$ and integer count thresholds for target false-bad allowances consisting of the symmetric target and 0.1, 0.3, or 0.5, while enforcing the calibrated false-clean cap. The conditional test cost is exactly $r(a + 1)$ module calls. It is a fixed-duration sampling service. The compiler uses confidence-upper false-clean probabilities with fitted false-bad probabilities; cost optimality for the unknown channel is not inferred from calibration coverage.

Execution and restoration. Two settings put clean prior mass 0.2 or 0.8; the remaining mass is uniform over seven fault locations. There are 400 tasks per setting. Controller order is randomly permuted within each task to reduce systematic timing-order confounding. Every baseline reads the same task and check-stream seeds. A rollback to k restores the suffix source files and the cached assumption record, then executes two passes over the full 128-row tolerance suite for each regenerated module. A final independent full-prefix label determines whether the retained prefix was truly valid. Labeling and initial task setup are excluded from recovery timing; paid diagnostic tests, report computation, file restoration, and replay are included. All file writes are also recorded explicitly.

Retry repeats the last workflow action on its existing source without repairing earlier files. It performs 128 tests and fails on all persistent-fault tasks, yielding 384 errors in 800 trials. It is a lower-work but unreliable comparator, not a risk-matched competitor. Full restart regenerates every module. The outcome table reports errors for every method rather than dropping failures from cost averages.

Repairing the oracle is a strong comparator. The tolerance-aware checker uses the contract already employed by the independent labeler. For this clean family it has $\beta = 0$ exactly. A fresh random test of a prefix containing the sign mutation detects it with probability 0.25. We separately calibrate this bad-prefix detection probability using 6,000 reports per boundary, 168,000 module tests, and choose fixed batch sizes with a safe false-clean bound. This is a changed checker, so it is displayed separately from same-menu planning comparisons. It achieves mean work 723.05, versus 1232.10 for full compilation with the brittle checker, with two versus zero observed errors. Its tolerance-oracle/brittle-one-step wall-time ratio is 0.64, bootstrap interval $[0.62, 0.65]$. Reusing the same calibrated detection bounds with a stricter 10^{-3} cap gives 802.03 tests, zero observed errors, and true-model marginal risk 1.32×10^{-4} (versus 3.81×10^{-4} for the full brittle-checker policy). Its wall-time ratio is 0.71, interval $[0.69, 0.72]$. This tighter comparator separates better diagnostics from spending more allowed risk. It underscores that a sophisticated planner can be the wrong response to a fixable test oracle.

Timing uncertainty. Ratios are ratios of mean recovery times over the fixed two-case grid. We aggregate consecutive groups of twenty paired tasks, then resample twenty such blocks within each case for 3,000 deterministic bootstrap replicates. This preserves short-range within-block timing dependence. Full/brittle-one-step is 1.02 $[1.01, 1.02]$; depth-two/brittle-one-step is 1.02 $[1.00, 1.03]$. These intervals quantify within-run variability for the evaluated workload; they do not establish performance on other workloads. Restoration and test-execution overhead can dominate small reductions in arithmetic work. This is why the study reports both unit-test counts and measured time.

F SCALING, REUSE, AND VALIDATION

F.1 PLANNING MEASUREMENTS AND CONDITIONAL AMORTIZATION

The primary timing experiment compiles each method three times in each of eight settings and reports the median over these measurements. It includes creating the final executable policy, the finite EC^2 search, and all lookahead replanning nodes. It excludes independent experiment execution and optional certificate passes. All methods use the same service definitions. Absolute and relative timings can change with implementation choices; these measurements apply to the evaluated implementations and workload.

The larger scaling study fixes the representative clustered profile, clean prior 0.8, and noise 0.25. At $n = 64, D = 5, h = 2$, the compiler evaluates approximately 12.3 million pairs, with maximum retained frontier 93. Retaining few vectors after pruning therefore does not make generation cheap. The $n = 128, D = 5$ timeout is censored at 90 seconds; no completion time is imputed.

For a reference controller g, compute per-invocation saving $\Delta = C_g - C_f$, incremental planning time $T_f - T_g$, and conversion v seconds per resource unit. Conditional payback is

$$N_{\text{payback}} = \left\lceil \frac{\max(0, T_f - T_g) + T_{\text{extra}}}{v\Delta} \right\rceil \quad \text{when } \Delta > 0.$$

T_{extra} includes optional certification and calibration not shared with the reference. If $\Delta \leq 0$, there is no finite payback from that estimate. Using unrounded values for payback, at 64 states the one-step-specific saving is 0.45 units, not the smaller-grid average saving; it yields 47,022 invocations at $v = 10^{-3}$ s/unit. Against nominal, the same instance saves 9.39 units with 20.95 s incremental planning, giving 2,231. The nominal value is an ablation-based payback, whereas the one-step value is the relevant inexpensive-policy comparison.

F.2 REUSE UNDER BOUNDED PARAMETER CHANGES

For fixed policies f, g, let $g_j = C_j(g) - C_j(f)$. If $\mathrm{TV}(b, b') \leq e$, then

$$b'^T g \geq b^T g - e(\max_j g_j - \min_j g_j). \tag{16}$$

To prove it, subtract the midpoint of the coordinate range from g. Its maximum absolute value becomes half the range; apply $\|b - b'\|_1 \leq 2e$. This is an elementary bounded-expectation inequality. It permits cheap reevaluation of a stored policy and an opportunity estimate; it does not estimate e from unrepresentative deployment data. Statewise risk is prior-independent, while expected cost is not.

If only replay and query prices change, reprice the statewise expected usage vectors before recompiling. If report laws change, their errors and conditional costs must be recalibrated or robustly bounded. A fresh prior cannot fix a miscalibrated diagnostic. A persistent failure mode requires an episode-level model or bound as in Section 3.1. Authored production-fault samples must not be treated as independent and identically distributed draws from deployment without an explicit sampling argument.

F.3 VALIDATION AND MEASUREMENT LIMITS

Validation compares actual-noise compilation against exhaustive trees and checks lookahead limiting cases, safe-policy mixtures, code-label and restoration semantics, confidence endpoints, information lower bounds, and finite-mode compilation. Numerical reproducibility is distinguished from physical timing reproducibility: repeated timing measurements are expected to differ.

These mathematical and empirical checks do not replace independent author review, formal verification, or external evaluation of semantic agent behavior.