Learning Which Computations to Trust: Audited Solver Choice, Fallible Verification, and Drift

Brian Theory
AI-assisted research draft

13 November 2026 — isolated research revision 3

Classification: Technical methods note. Standard decision and concentration tools applied to a declared solver interface; no new general metareasoning principle or priority claim.

Abstract

An agent can use a model of its computations to choose a solver, verify its output, or invoke an exact fallback. We give a finite operational contract in which approximate solvers generate candidate witnesses, a trusted checker reveals correctness at a cost, and an exact solver repairs a failed candidate. A five-action deployment menu has an explicit cost frontier. Independent audited training gives a simultaneous calibration guarantee and a finite-sample excess-cost bound; a margin condition is required to compare certified policies with a known-reliability oracle. A finite-horizon Bayesian recursion distinguishes verification for the current answer from verification that also improves future solver choice. For independent errors with slowly changing means, a sliding audit window has an explicit sampling-error plus drift-bias bound. Executable partial-constraint solvers and exact finite-state calculations illustrate these results, including cases where additional computation is redundant or current verification is costly but future learning makes it worthwhile. An extended contract prices controller overhead, derives residual risk for fallible checking, and gives a sparse scheduled noisy-audit bound with a warmup bill. A finite hidden-state recursion handles noisy learning and switching costs. Direct gold-label calibration also bounds invalid acceptance without a known checker channel. A frozen public SAT study and a disclosed larger-instance amendment measure complete pipeline and setup costs. Risk-constrained selection chooses the fixed exact solver, and a mature exact baseline further strengthens the negative result. The work is a limited synthesis of established metareasoning, algorithm selection, and risk calibration. It establishes neither universal architectural superiority nor subjective introspection.

1 The computational question and its antecedents

Metacognitive control is useful only if a model of computation changes which computation is performed or how its output is used. A verbal confidence label without consequences is not the object of study. Here, the controller learns how often particular algorithms produce invalid candidate witnesses on a declared task distribution, and uses that knowledge to allocate verification and fallback computation.

Hay, Russell, Tolpin, and Shimony formalize computation selection as a metalevel decision problem with explicit computational outcomes and costs [6, Definitions 1–3 and Theorems 4–5]. Algorithm portfolios already learn which solver to use and how to schedule alternatives; Kadioglu and colleagues provide a concrete selection/scheduling framework and matched SAT benchmarks [10, Sections 2–4]. Neither choosing an algorithm from experience nor the expected value of computation is introduced here.

The nearest B-Theory note, *When Checking Takes Time*, version 1.0 of 9 November 2026, gives an exact two-probe stopping example with a changing external mechanism and known noise [2]. *Finite Noisy Audits with Learned Baselines*, version 1.0, already prices observations and separates learned baselines from audit evidence [3]. The change here is the provenance of the outcome: it is produced by an executed candidate-generation algorithm, and correctness is supplied only by a paid checker. The mathematical reduction still uses ordinary statistics. This provenance supports an operational interpretation as computation control; it does not make a familiar concentration inequality a new metacognition theorem.

The results below should be read as a methods note and a reproducible starting model. Their strongest contribution to the portfolio is a precise distinction between current answer quality, the cost of verification, and information about future computation quality. General policies that combine arbitrary solver traces, nonstationarity, and scarce verification remain unresolved.

2 Operational contract

Tasks arrive in observed regimes $z \in \{1, \dots, J\}$. In the stationary part, independent instances x are drawn from an unknown fixed distribution D_z. Each instance has a nonempty finite witness set

$$\mathcal{W}(x) = \{w : V(x, w) = 1\}.$$

Two supplied randomized algorithms S_1, S_2 read x, execute their prescribed partial computation, and output a candidate w_i. The complete checker V computes whether that candidate is valid. An exact algorithm E returns a valid witness. Algorithms, checker semantics, and action costs are common knowledge; their population error rates

$$p_{iz} = \mathbb{P}_{x \sim D_z, \text{solver coins}}\{V(x, S_i(x)) = 0\}$$

are initially unknown. Solver coins are fresh across tasks. Errors from different solvers on the *same* task may be dependent.

Running solver i costs $c_i \geq 0$, checking a returned candidate costs $v \geq 0$, and invoking the exact solver costs $r \geq 0$. Costs are normalized task-loss units, not measured seconds. An invalid terminal witness incurs additional loss one. An exact fallback always terminates within the stated task deadline. The instance and target do not change during computation. No unpriced evaluator reveals whether an unverified answer was correct.

For the first result the controller has five permitted deployment policies:

Policy	Executed operations	Expected task loss
U_i	run S_i, return candidate	$c_i + p_{iz}$
V_i	run S_i, check, use E only if invalid	$c_i + v + rp_{iz}$
E	run exact solver	r

There are two policies of each indexed type. Verified policies have zero terminal error under the trusted-checker and exact-fallback assumptions. Their running cost varies with candidate validity. The menu excludes inspecting extra solver traces, combining candidates, or using input features beyond z to choose the menu entry. Those are meaningful extensions rather than secretly available privileges.

This model has genuine computational acts: the candidate is computed from the task instance, the verifier executes a correctness test, and the fallback constructs a solution. Nevertheless, retaining

only the error indicator gives a Bernoulli statistical interface. Our theorems concern that interface, not a separation from every possible algorithm on x. An unrestricted compiler could encode the same choice mapping, and a different exact algorithm could have a different cost. In Sections 2–8, fixed code, controller memory, and exact real arithmetic are unpriced; later sections extend the accounting and report bounded implementation measurements. The comparisons make no asymptotic implementation claim.

3 A complete known-reliability menu frontier

Write $L_z(d; p)$ for the loss in the table and

$$L_z^*(p) = \min\{r, c_1 + p_{1z}, c_2 + p_{2z}, c_1 + v + rp_{1z}, c_2 + v + rp_{2z}\}. \tag{1}$$

Proposition 3.1 (Elementary decision frontier)**.** *Equation* (1) *is the minimum expected loss in the stated five-policy menu. Randomizing among menu entries cannot reduce it. Verification after a chosen solver reduces current-task loss relative to returning its candidate exactly when*

$$v < (1 - r)p_{iz}. \tag{2}$$

At equality the two choices tie. If $r \geq 1$ and $v \geq 0$, verification with this fallback cannot strictly improve the current-task objective.

Proof. Condition on candidate validity. An unverified candidate fails with probability p_{iz}. A verified candidate incurs the exact-solver charge only on that failure event and is always valid at termination. These give the table. A mixture of entries is a convex combination of their losses and cannot be below their minimum. Subtracting the unverified loss from the verified loss gives $v - (1 - r)p_{iz}$. □

The criterion is population averaged within a regime. It is not a per-instance correctness guarantee. If a current candidate has an independently justified conditional error probability q, the same algebra uses q instead of p_{iz}. Substituting a marginal error rate for an arbitrary instance's conditional error can be badly miscalibrated.

A simple stopping boundary is also immediate. If the current candidate has error probability q, any further computation with deterministic cost exceeding q cannot improve a one-task zero–one objective even if it removes all error. This is a special case of the established value-of-information bound, not a new principle. Future tasks change that comparison, because a paid check can teach the controller about its solver.

4 Learning reliability from an audited training sample

For each regime, collect n independent training tasks and run *both* solvers and the full checker on each candidate. This full-information audit costs

$$C_{\text{train}} = nJ(c_1 + c_2 + 2v). \tag{3}$$

The exact fallback is not run during training. Training tasks have no additional deployment-success obligation; if they did, that cost would have to be added. Let $Z_{ijz} \in \{0, 1\}$ indicate the error of solver i on task j in regime z, and let $\widehat{p}_{iz} = n^{-1} \sum_j Z_{ijz}$.

Lemma 4.1 (Standard bounded-sample concentration)**.** *For n independent variables in $[0,1]$ with means μ_j, their mean $\widehat{\mu}$ satisfies*

$$\mathbb{P}\{|\widehat{\mu} - n^{-1}\textstyle\sum_j \mu_j| > \eta\} \le 2e^{-2n\eta^2}.$$

Identical distributions are unnecessary.

This is Hoeffding's inequality [7, Theorem 2]. To make the dependence explicit, for a bounded variable the centered log moment-generating function has second derivative equal to a tilted variance, at most $1/4$. Its value and derivative at zero vanish, so it is at most $\lambda^2/8$. Multiplying these bounds over independent variables, applying Markov's inequality, and minimizing at $\lambda = 4\eta$ for the average gives $e^{-2n\eta^2}$ for either tail; add the two tails. The variance bound follows from $\mathrm{Var}(X) \le \mathbb{E}(X - 1/2)^2 \le 1/4$ for any law on $[0,1]$, including the tilted law.

Put $K = \max\{1, r\}$ and

$$\eta_n = \sqrt{\frac{\log(4J/\delta)}{2n}}, \qquad 0 < \delta < 1. \tag{4}$$

For each z, let $\widehat{d}_z$ minimize $L_z(d; \widehat{p})$ over the five entries, with a fixed public tie rule.

Theorem 4.2 (Simultaneous calibration and menu regret)**.** *With probability at least $1 - \delta$ over the training sample, simultaneously for all i, z,*

$$|\widehat{p}_{iz} - p_{iz}| \le \eta_n,$$

and simultaneously for all regimes,

$$L_z(\widehat{d}_z; p) - L_z^*(p) \le 2K\eta_n. \tag{5}$$

The same event supports every predeclared mixture of the regimes at deployment.

Proof. Apply Lemma 4.1 to each of the $2J$ empirical error rates. A union bound gives failure probability at most $4Je^{-2n\eta_n^2} = \delta$. Independence between the two solvers on the same task is not used. On the good event, every menu loss changes by at most $K\eta_n$, because its dependence on a single error rate has slope zero, one, or r. If d_z^* minimizes true loss, then

$$L_z(\widehat{d}_z; p) \le L_z(\widehat{d}_z; \widehat{p}) + K\eta_n \le L_z(d_z^*; \widehat{p}) + K\eta_n \le L_z(d_z^*; p) + 2K\eta_n.$$

Average these simultaneous inequalities with any fixed nonnegative regime weights summing to one. □

This is a finite-action empirical-risk comparison using a standard uniform bound. The known-reliability oracle is privileged and pays no calibration charge. For H deployments the learner's good-training-event expected total cost is bounded by $C_{\text{train}} + H(L^* + 2K\eta_n)$ for the chosen regime mixture. It is incorrect to omit C_{train} or to describe a high-probability conditional bound as a deterministic bill. An unconditional bound adds at most $B\delta$ per deployment, where $B = \max_i\{c_i + 1, c_i + v + r, r\}$ bounds possible regret.

4.1 A certified-error constraint needs a margin

Suppose terminal error must be at most $\alpha \in [0,1]$ in each regime. Permit U_i only when $\widehat{p}_{iz} + \eta_n \le \alpha$; keep V_1, V_2, E available. Select the cheapest estimated loss from this certified menu.

Corollary 4.3 (Risk certification with a comparator margin)**.** *On the same event, every selected policy has terminal error at most α. Its loss is at most $2K\eta_n$ above the best of the exact/verified policies and the unverified policies satisfying $p_{iz} \le \alpha - 2\eta_n$.*

Proof. Every admitted unverified policy has $p_{iz} \le \widehat{p}_{iz} + \eta_n \le \alpha$; all other entries have zero terminal error. If $p_{iz} \le \alpha - 2\eta_n$, then $\widehat{p}_{iz} + \eta_n \le p_{iz} + 2\eta_n \le \alpha$, so that comparator is admitted. Apply the same two-error comparison as in Theorem 4.2. □

The loss comparison deliberately excludes an unverified oracle action exactly on the risk boundary unless it is independently certified. Uniformly matching a boundary oracle from finite data would require a stronger argument. Certification is a property conditional on the training event and the assumed unchanged deployment distribution, not a guarantee that a particular emitted witness is correct.

5 Verification can have value for future tasks

An alternative contract learns during deployment. For one regime, suppose the error rates are independent latent variables $p_i \sim \mathrm{Beta}(a_i, b_i)$. Conditional on the rates, solver errors on successive tasks are independent; unverified candidates supply no correctness feedback. A verified error increments a_i and a verified success increments b_i. No update follows an exact-only action. Let $z = (a_1, b_1, a_2, b_2)$, $\bar{p}_i = a_i/(a_i + b_i)$, and z_i^-, z_i^+ denote failure and success updates.

Proposition 5.1 (Standard finite-horizon metalevel dynamic program)**.** *With h tasks left, the optimum Bayes total cost within this online menu is $V_0(z) = 0$ and*

$$V_h(z) = \min\Big\{ r + V_{h-1}(z), \\ \min_i\{c_i + \bar{p}_i + V_{h-1}(z)\}, \\ \min_i\{c_i + v + r\bar{p}_i + (1-\bar{p}_i)V_{h-1}(z_i^+) + \bar{p}_i V_{h-1}(z_i^-)\}\Big\}. \tag{6}$$

Proof. The beta posterior follows by multiplying its density by p_i after a verified failure or by $1 - p_i$ after a success. Predictive failure probability is $\bar{p}_i$. Condition on the first menu choice and, for verification, its two possible outcomes. The immediate expected costs are exactly those of Proposition 3.1. Posterior parameters contain all retained information relevant to future task errors in this stipulated exchangeable model. Backward induction over the finite horizon proves the recurrence and permits a deterministic minimizer at each state. □

This recursion is not a new form of rational metareasoning; it makes one feedback assumption explicit. If unverified answers later receive trustworthy labels, the second line must also branch and learn. If errors depend on retained solver state or correlated task families, the beta counts are not sufficient.

The reproducible example takes $c_1 = .02, c_2 = .05, v = .06, r = .6$, an unknown solver-1 error with prior $\mathrm{Beta}(1, 9)$, and a known solver-2 error .1. At the initial mean, unverified solver 1 costs .12 and verified solver 1 costs .14. Thus checking is worse for the isolated current task. The supplied

exact rational recursion evaluates horizons $1, 2, 5, 10, 20$ and records how future information changes the initial choice. This is a Bayes planning example under a supplied prior, not an empirical claim about general reasoning quality.

Horizon h	Optimal expected cost	First action	Always U_1 cost
1	0.120000	U_1	0.12
2	0.240000	U_1	0.24
5	$824/1375 \approx 0.599273$	V_1	0.60
10	$12877/11000 \approx 1.170636$	V_1	1.20
20	$12364281/5434000 \approx 2.275355$	V_1	2.40

Table 1: Exact finite-horizon Bayesian computation for the stated prior and costs. The information gathered by initial verification has sufficient future value to change the optimal first action at the displayed longer horizons. The reference policy always uses the same solver without feedback. This table does not assert a horizon threshold between the evaluated cases.

6 Slow drift and stale solver calibration

Now fix a regime and allow independent error indicators $Z_{i,t}$ with changing means $p_{i,t}$. Assume the deterministic rate constraint

$$|p_{i,t+1} - p_{i,t}| \le \rho.$$

For this extension, each task first commits to and completes one deployment-menu action using only earlier audit labels. A separate audit then reruns both solvers with fresh private coins on the saved current instance and checks both new candidates. These correctness labels are disclosed only after the terminal deployment answer, so they cannot repair a nominally unverified answer on that task. Both reruns and both checks are charged, even when they duplicate production work: the fixed additional per-task audit cost is

$$C_{\text{audit}} = c_1 + c_2 + 2v.$$

The cost for choosing entry d is therefore $L(d; p_t) + C_{\text{audit}}$. Comparators in the drift result use this same audit protocol. No computational reuse or unpriced current feedback is assumed. Before task $t > w$, estimate from the preceding w audits,

$$\widehat{p}_{i,t} = \frac{1}{w} \sum_{j=1}^{w} Z_{i,t-j}.$$

Theorem 6.1 (A fixed-window drift bound; standard concentration specialization)**.** *For a predeclared horizon of at most T evaluated tasks and window w, with probability at least $1 - \delta$, simultaneously for the two solvers and all evaluated times,*

$$|\widehat{p}_{i,t} - p_{i,t}| \le \underbrace{\sqrt{\frac{\log(4T/\delta)}{2w}}}_{\textit{sampling fluctuation}} + \underbrace{\rho \frac{w+1}{2}}_{\textit{staleness bias}} =: e_w. \tag{7}$$

Plug-in menu selection then has current expected deployment-menu excess loss at most $2Ke_w$ at every such time on this event. Adding the common audit charge to both comparator costs preserves that difference; it does not remove the audit expenditure from total cost.

Proof. The mean of the window estimator is the average of its w possibly unequal Bernoulli means. Lemma 4.1, applied at each fixed window, and a union bound over at most $2T$ windows bound deviations from those averages by the first term. Overlap between windows does not affect the union bound. The deterministic drift condition gives

$$\left| \frac{1}{w} \sum_{j=1}^{w} p_{i,t-j} - p_{i,t} \right| \leq \frac{\rho}{w} \sum_{j=1}^{w} j = \rho \frac{w+1}{2}.$$

Combine the two inequalities and repeat the loss comparison from Theorem 4.2. □

The theorem is not valid for an adaptively chosen window without accounting for that selection, nor does it infer ρ from the same data. Its formula quantifies the tension between a long window's reduced sampling error and increased staleness. Minimizing the two variable terms over real w gives the usual scale $w \asymp \{\log(T/\delta)/\rho^2\}^{1/3}$ when $\rho > 0$, before integer and horizon constraints. No minimax optimality is claimed. Section 10 treats deterministic sparse noisy audits; adaptive sparse verification, abrupt changes, and input-dependent nonstationarity remain unresolved.

7 Executed finite constraint-solving study

The illustration uses witnesses $w \in \{0, \ldots, 15\}$. An instance contains two membership predicates C_1, C_2 with a nonempty intersection. Solver i enumerates candidates satisfying C_i and chooses one uniformly; it defers the other constraint. The full checker evaluates both. The exact fallback scans candidates and returns the first satisfying both. A uniformly rotated labeling creates sixteen instances per regime. Every solver, verifier, and comparator receives the same instance; no hidden task label is exposed to one controller alone.

The three regime families have candidate-set sizes and intersection sizes

Regime	$\lvert C_1 \rvert$	$\lvert C_2 \rvert$	$\lvert C_1 \cap C_2 \rvert$	(p_1, p_2)	Oracle menu choice
A	10	15	9	$(.1, .4)$	U_1, loss .12
B	15	10	9	$(.4, .1)$	U_2, loss .15
D	5	5	3	$(.4, .4)$	V_1, loss .32

The error formula is $p_i = 1 - |C_1 \cap C_2|/|C_i|$. The complete task sets, rotations, and solver code are supplied. Enumeration checks 48 task instances and 96 solver error laws. A separate seeded execution runs each solver 20,000 times per regime, totaling 120,000 trials, as an implementation check. This is synthetic computation, not a benchmark showing a practical speedup.

The figure's learning curves are finite sums over the two binomial training error counts, with $n \in \{1, 2, 5, 10, 25, 50, 100, 200\}$. Independent solver seeds and the constant regime-specific error rates justify the product of the two count laws in this example; the general guarantee does not require within-task independence. The sums are evaluated in floating point and saved with the known-rate comparator. They describe expected *deployment* cost; the separate training cost is recorded rather than hidden in the plot.

The supplied costs deliberately isolate the decision problem. On such small explicit sets, another implementation could directly compute the intersection and cheaply solve every task. Therefore the illustration establishes a checkable solver-feedback contract, not a computational advantage over all programs. Real algorithm portfolios must measure feature extraction, checker, solver, switching, and controller overhead on a suitable task distribution.

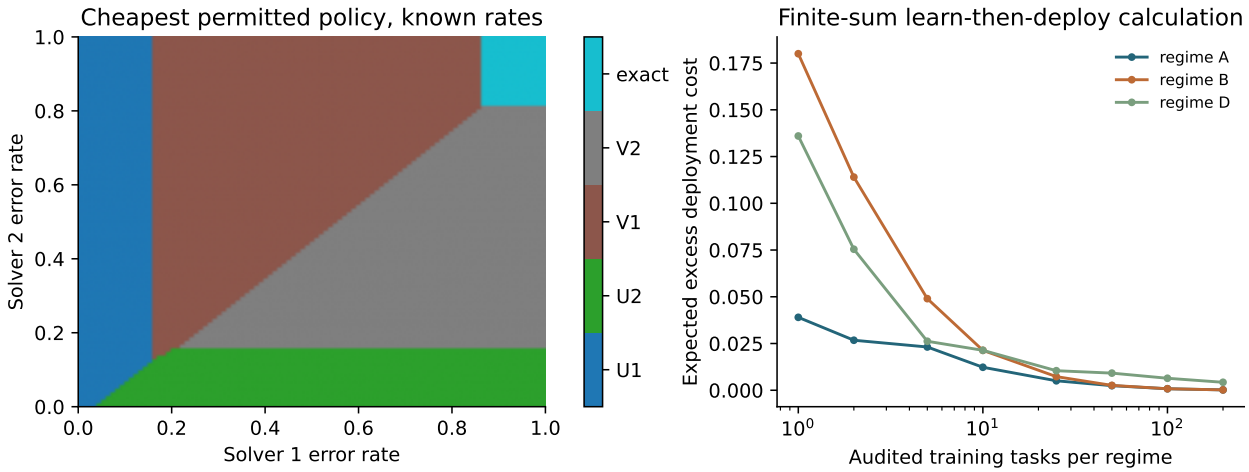


Figure 1: Original calculations for the stated prices. Left: the cheapest permitted policy for each known pair of error rates (U: unverified; V: verified with exact rescue). Right: finite-sample expected deployment excess cost of the plug-in selector in three constructed regimes. The cost axis is stipulated loss units, not runtime. Editable SVG, data, and generation code accompany the PDF.

8 Counterexamples and limits of inference

Biased verification. Let half the tasks be easy with solver error zero and half hard with error .8. Auditing only easy tasks estimates error zero even with unlimited data, while the overall error is .4. Theorem 4.2 does not apply: training is no longer representative. Adaptive auditing needs an explicit missing-feedback correction, randomized sampling design, or a valid conditional model.

Re-running a deterministic computation. If a solver sees the same input and retains the same seed or deterministic state, it returns the same candidate. Repeating it without a new check adds no information about validity and consumes positive cost. Multiplying its marginal error rates as if repetitions were independent creates fictitious confidence. New independent seeds alone also do not imply conditional independence of failure for arbitrary shared task difficulty.

Marginal calibration is not introspective certainty. A regime-average error rate can be accurate while errors are concentrated on one narrow family of inputs. These results calibrate population rates, not confidence for every answer. Instance-level reliability needs a richer observation model and held-out conditional validation.

A drifting world during deliberation. The task instance remains fixed until a final witness is returned. If the target changes while a solver runs, runtime changes the target distribution and verification can become stale. A common deadline or a clearly indexed target is required before comparing policies; the earlier two-probe B-Theory note studies a different timing contract. Its conclusions should not be imported without a new derivation.

The checker is a trusted resource. If V can be wrong, verified policies no longer have zero terminal error. Shared bugs between candidate generation and checking can defeat repeated agreement. A genuinely uncertain checker requires a second reliability model, identifiable references, and a cost for validating it. D03's evidence-validity problem then becomes relevant; the later

known-channel extension does not identify an unknown checker or resolve an infinite regress of verification.

9 Fallible checking and controller costs

Sections 2–8 retain the trusted-checker contract. We now change it explicitly. These additions are finite-model specializations, not a claim that realistic verification or general metareasoning has been solved. Let $Z_i = 1$ denote an invalid candidate and $Y_i = 1$ denote rejection by a fallible checker. Assume known conditional rates

$$a_i = \mathbb{P}(Y_i = 0 \mid Z_i = 1), \qquad b_i = \mathbb{P}(Y_i = 1 \mid Z_i = 0), \qquad \gamma_i = 1 - a_i - b_i.$$

These rates must hold for the deployed solver, candidate population, regime and time, including each latent state below. Accuracy on an unrelated checker test set does not suffice. A rejection invokes the exact fallback; an acceptance returns the candidate. Call this policy W_i, reserving V_i for the earlier trusted checker. Exact fallback remains an assumption.

Let $h_{U_i}, h_{W_i}, h_E \geq 0$ be fixed deterministic controller charges in the same loss units. They can include dispatch, feature extraction, memory updates and amortized offline construction, if billed once. Memory, time and energy need declared conversions or separate constraints; arbitrary resource units cannot be added.

Proposition 9.1 (Fallible-check frontier)**.** *Writing $q_i = b_i + \gamma_i p_i$, the action losses and terminal errors are*

Action	*Expected loss*	*Terminal error*
U_i	$c_i + h_{U_i} + p_i$	p_i
W_i	$c_i + v_i + h_{W_i} + rq_i + a_i p_i$	$a_i p_i$
E	$r + h_E$	0

Their minimum is the five-entry menu optimum. Checked action W_i strictly improves on U_i exactly when

$$v_i + h_{W_i} - h_{U_i} + rb_i < [1 - a_i - r\gamma_i]p_i. \tag{8}$$

Proof. The joint masses for $(Z_i, Y_i) = (0,0), (0,1), (1,0), (1,1)$ are $(1-p_i)(1-b_i), (1-p_i)b_i, p_i a_i, p_i(1-a_i)$. Fallback has probability q_i; invalid acceptance has probability $a_i p_i$. Summation and subtraction give the claims; randomization remains a convex combination. □

Equal controller charges and $a = b = 0$ recover the original frontier. False rejection pays for unnecessary fallback; false acceptance leaves residual error. A checker with $a = 1, b = 0$ always accepts and provides neither repair nor information. Repeating an identical shared bug does not fix it.

9.1 Identification and certified decisions

Rejection frequencies alone do not identify both solver and checker reliability. The models $(p, a, b) = (1/10, 0, 0)$ and $(1/5, 1/2, 0)$ both give $q = 1/10$, but checked terminal risks are zero and $1/10$. Even infinite independent labels with this identical law do not distinguish the models through this interface. Known noise correction and identifying the noise matrix are different problems [11]. Gold references or additional identifiable structure must be supplied and priced.

Suppose known channels and clipped estimates satisfy $|\widehat{p}_i - p_i| \le e$. Put

$$M = \max\{1, \max_i |r\gamma_i + a_i|\}, \qquad u_i = \min\{1, \widehat{p}_i + e\}.$$

For an error limit α, admit U_i only when $u_i \le \alpha$, W_i only when $a_i u_i \le \alpha$, and always admit E. All admitted actions are safe on the confidence event. Write $m_{U_i} = 1, m_{W_i} = a_i, m_E = 0$. A comparator whose true risk is at most $\alpha - 2m_d e$ is admitted because $u_i \le p_i + 2e$. Zero-slope risks need no margin.

If estimated controller charges have uniform error at most ϵ_c and an implementation returns a certified action within $\zeta \ge 0$ of the smallest estimated certified-menu loss, its excess over the best margin-admitted comparator is at most

$$2Me + 2\epsilon_c + \zeta. \tag{9}$$

Every estimated action loss differs from truth by at most $Me + \epsilon_c$; adding this error on both sides of the approximate minimization proves the statement. This does not supply runtime confidence bounds or a certified numerical optimizer for free. With switching costs, a one-step comparison is from the same previous-action state, not against a globally optimal switching sequence.

Estimated a, b cannot be treated as exact here. Given an independently justified simultaneous confidence set Θ_t for (p_i, a_i, b_i), the rule $\sup_{\theta \in \Theta_t} R_d(\theta) \le \alpha$ guarantees safety whenever Θ_t covers truth. Constructing, computing and paying for that set remains part of the calibration experiment; no learned-checker experiment is claimed.

10 Sparse scheduled noisy audits under drift

In this contract $p_{i,t}$ are deterministic means with $|p_{i,t+1} - p_{i,t}| \le \rho$. Current deployment outcomes must have these laws conditional on earlier disclosed audit history; independent tasks and fresh coins suffice. Each scheduled audit separately reruns both solvers and noisy checks after the production answer. Its charge is $C_A = c_1 + c_2 + v_1 + v_2 + h_A$, including audit processing. Assume known time-invariant channels with $\gamma_i > 0$, independent audit labels across tasks for each solver, and deterministic audit times fixed before observing labels. Within-task dependence between solvers is permitted. This protocol uses only audit-copy labels, deliberately ignoring production checker feedback.

Before evaluated task t, the last n audits have times $\tau_{t,j} < t$. Define

$$\widehat{p}_{i,t} = \Pi_{[0,1]}\left(\frac{\overline{Y}_{i,t} - b_i}{\gamma_i}\right), \qquad A_t = \frac{1}{n}\sum_{j=1}^{n}(t - \tau_{t,j}).$$

The window and horizon of at most T evaluated tasks are predeclared. Purchase sufficient prehistory audits or begin the guarantee after warmup.

Theorem 10.1 (Noise and audit-age bound)**.** *For $0 < \delta < 1$, with probability at least $1 - \delta$, simultaneously over two solvers and at most T evaluated tasks,*

$$|\widehat{p}_{i,t} - p_{i,t}| \le e_{i,t} := \min\left\{1, \frac{1}{\gamma_i}\sqrt{\frac{\log(4T/\delta)}{2n}} + \rho A_t\right\}. \tag{10}$$

Plug-in menu selection has conditional expected deployment-loss excess at most $2M \max_i e_{i,t}$. Approximation and certification use the modified rules above.

Proof. Each label mean is $b_i + \gamma_i p_{i,\tau}$. Hoeffding and a union bound control every window's deviation from its own mean by the square-root term before inversion. Inversion divides this fluctuation by γ_i and maps the expected rejection average to the average of past p_i values. Its distance to current $p_{i,t}$ is at most ρA_t. Projection cannot increase distance to $p_{i,t} \in [0,1]$ and the distance is at most one. Apply the loss slopes and conditional deployment-law assumption. □

The drift term is not divided by γ_i. Known negative γ_i is also invertible, with $|\gamma_i|$ in the sampling denominator. At $\gamma_i = 0$, labels are uninformative about p_i. Checker drift invalidates this fixed-channel derivation.

With period k, the latest audit age is $d_t \in \{1, \dots, k\}$ and $A_t = d_t + k(n-1)/2 \le k(n+1)/2$. For common γ and $\rho > 0$, balancing the variable terms gives

$$n \asymp \{\log(T/\delta)/(\gamma^2 \rho^2 k^2)\}^{1/3}.$$

This minimizes a bound, not minimax regret. Choosing n, k deterministically from design parameters is allowed; adapting them to labels needs another proof.

For common learner-only controller charge h, a known-rate oracle paying neither controller nor audits, n warmup audits and at most $\lceil T/k \rceil$ deployment audits, the good-event difference of predictable expected total losses is bounded by

$$2M \sum_{t=1}^{T} \max_i e_{i,t} + C_A\{n + \lceil T/k \rceil\} + Th + C_{\text{setup}}. \tag{11}$$

Here base menu losses exclude that separately charged common h; C_{setup} includes offline design and channel calibration. The statement concerns expected stage losses on a confidence event, not realized bills, a fair-resource oracle, or a runtime deadline guarantee. Unconditional expectation needs a bounded bad-event remainder. Only costs actually shared by a comparator cancel.

The design calculation uses $T = 10^4, \delta = .05, \rho = 10^{-4}, \gamma = .85, C_A = .19, M = 1, h = .001$ and optimizes $1 \le n \le 500$ separately for $k = 1, 2, 5, 10, 25$. With known channels and $C_{\text{setup}} = 0$, period 2, window 500 gives radius .18726 and upper bound .48002 per deployment; period 1, window 500 gives .52492. The bound is loose and certifies no practical improvement. Warmup costs nC_A, takes nk time steps, and requires the drift law over that prehistory. This is a deterministic bound-design calculation, not a simulated optimal audit policy.

11 Bayesian noisy feedback and hidden drift

This is a separate stochastic contract. A finite hidden state θ_t has known transition matrix P applied after each task. Conditional on the state, fresh error probabilities are $p_i(\theta_t)$ and the checker channel is stable. The belief π precedes computation. In the abstract controller interface only a checked outcome Y supplies learning: candidate identity, input features, runtime, fallback witness and realized error penalties reveal no further information. An implementation exposing such signals needs a changed observation kernel.

Write $\bar{p}_i = \sum_\theta \pi(\theta) p_i(\theta)$, $q_i(\theta) = b_i + \gamma_i p_i(\theta)$ and $\bar{q}_i = \sum_\theta \pi(\theta) q_i(\theta)$. Bayes' rule gives

$$\pi_i^y(\theta) = \frac{\pi(\theta) q_i(\theta)^y [1 - q_i(\theta)]^{1-y}}{\bar{q}_i^y (1 - \bar{q}_i)^{1-y}}.$$

A zero-probability branch contributes zero and needs no posterior definition. Let ℓ be the previous menu entry, $s(\ell, d) \ge 0$ a switching charge (initially zero), and $g_d(\pi)$ the noisy-check table's immediate loss using posterior means.

Proposition 11.1 (Finite-horizon belief recursion)**.** *With $B_0 = 0$, the optimum within this interface is*

$$B_h(\pi, \ell) = \min_d \{g_d(\pi) + s(\ell, d) + F_{h-1}(d, \pi)\},$$
$$F_{h-1}(d, \pi) = \begin{cases} B_{h-1}(\pi P, d), & d \in \{U_1, U_2, E\}, \\ (1 - \bar{q}_i) B_{h-1}(\pi_i^0 P, d) + \bar{q}_i B_{h-1}(\pi_i^1 P, d), & d = W_i. \end{cases}$$

Proof. Condition on the first action and its available observation. Update the current belief before applying the state transition. Without observations the prediction is πP. Remembering ℓ accounts for switching costs. Backward induction gives optimality on the finite history tree. A finite hidden-state space does not imply a finite belief space for an unbounded horizon. □

Noisy rejection cannot increment the original beta failure count: its likelihood is $b + \gamma p$, not p. Multiplying a beta density generally produces a mixture. With $a = b = 1/2$, either label leaves the posterior unchanged; the code tests this counterexample. This Bayesian model is not automatically covered by Theorem 10.1; posterior jumps do not supply its deterministic-mean assumptions.

Exact examples use solver-1 rates $\{.02, .18\}$ with equal initial weights, known solver-2 rate .1, $a = .1, b = .05$, $c_1 = .02, c_2 = .05, r = .6$, checker prices .06, .02, symmetric state-flip probabilities $0, .02$, and switch charges $0, .002$. These priors differ from the original beta example, so differences cannot be attributed to noise alone. Eighteen exact cases at horizons $1, 5, 8$ record fractions, first actions, state counts and planning times. Policy-tree evaluation checks achieved values using the selected Bellman policy; it is not another optimality proof. A separate specialist implementation independently reconstructs the three displayed horizon-eight optima by exact alpha-vector policy-tree enumeration with rational lower-envelope pruning, without reusing the posterior recursion.

Horizon	Checker price	Flip probability	Optimum loss	First action
8	.06	0	.960000000	U_1
8	.02	0	.959498712	W_1
8	.02	.02	.960000000	U_1

These displayed rows have zero switching charge; the full table includes the positive-charge cases. The middle value is exactly 47974935622137185777/50000000000000000000. Its first checked action costs .131 versus .12 unverified. Future information gives a small net gain, while hidden drift removes it in the displayed comparison. No universal horizon threshold or robustness claim follows.

12 Audit evidence, measured costs, and research limits

All five regenerated original data files match their delivered bytes. The earlier review closure's source and code hashes match the delivered artifacts. A separate new specialist implementation reproduces all five original rational Bayesian values and checks 27,951 rational noisy-channel cases. The new author implementation separately checks 14,641 joint-event cases. Printed proofs support general claims; finite checks support implementation consistency. New reviews are internal AI reviews, not human peer review.

The original five-task advantage over always U_1 is

$$\frac{3}{5} - \frac{824}{1375} = \frac{1}{1375}.$$

An additional uniform per-task controller charge relative to that fixed baseline erases this particular gain at $1/6875 \approx 1.45 \cdot 10^{-4}$. A one-time additional charge has break-even 1/1375. These statements concern the specified policy comparison, not every possible policy. A common charge can leave action rankings unchanged yet erase an advantage over a cheaper fixed implementation.

The local timing experiment uses all 48 prepared original instances, three warmup batches and 21 measured batches of 4,800 calls, with shuffled operation order and raw nanosecond records. It measures partial candidate generation, checking, scan-based exact solving, direct exact intersection, floating menu calculation and a compiled lookup. Direct intersection is faster in this recorded environment than either candidate generation or floating menu calculation. Thus the tiny tasks support no portfolio speedup claim. They do not turn synthetic loss prices into realistic costs. Dispersion across batches is not a confidence interval across workloads or machines. Cold input construction and hard deadlines are excluded. Exact rational planning times and state counts are recorded separately. Compiling a policy can save online planning but incurs construction, memory, lookup and model-obsolescence costs.

Charging selection overhead is established practice: SATzilla includes feature and presolver time and a timeout fallback [16, Sections 4.2–4.3]. Costly-feedback learning and paid multi-arm observations already appear in Tucker et al. [15], Seldin et al. [14]; current contextual work also includes observation expenditure [5]. Smooth nonstationary bandits have stronger results under different time/feedback assumptions [9]. No regret-rate transplant or priority claim is made. The increment here is an auditable repaired-output loss, known noisy-check channel, sparse scheduled-audit certificate, and exact counterexamples and implementation checks.

Revision 2 left learned-checker calibration and public-task cost validation open. Sections 13–15 address direct gold calibration and bounded public-corpus validation; adaptive biased audits, unknown abrupt drift and representative solvers under a common deadline remain substantive blockers. Gold labels and exact fallback reliability remain paid resources.

Share generic concentration and noisy-channel methods with D03/D04. D03 owns physical evidence freshness; D05 owns recurrent resource viability; D07 owns memory/coding; D10 owns a specified physical energy boundary. D08 owns solver-produced candidates, checker feedback, repair choices and controller costs. This revision makes no joint theorem about those manuscripts and edits none of them. The operational capability does not establish consciousness, subjective introspection or intrinsic superiority of explicit self-model terminology.

13 Gold-label calibration without a known checker channel

The known-channel assumption of Section 9 is avoidable when calibration pays for trusted truth. This changes the information contract: rejection-only observations still do not identify residual error. Fix m solver/checker pairs and an unchanged iid task population. On calibration draw j, let Z_{ij} indicate an invalid candidate, as determined by paid gold checking, and let Y_{ij} indicate rejection by the deployed fallible checker. Set $U_{ij} = Z_{ij}(1 - Y_{ij})$. Arbitrary within-task dependence is allowed. Draws, including solver and checker randomness, are independent across tasks. Exact repair on rejection gives

$$p_i = \mathbb{E}Z_i, \quad q_i = \mathbb{E}Y_i, \quad u_i = \mathbb{E}U_i, \qquad L(W_i) = c_i + v_i + h_i + rq_i + u_i.$$

The terminal error of W_i is u_i. Neither conditional checker ratios nor a lower bound on invalid-candidate prevalence is needed. Rare invalid candidates can make a conditional false-acceptance estimate unstable while the joint event remains directly estimable.

Theorem 13.1 (Finite-menu joint-event calibration)**.** *Let m, n be positive integers, $0 < \delta < 1$ and $0 \le \alpha \le 1$. Assume deterministic nonnegative prices, known r, and exact repair, with unit terminal-error loss. Let $K = \max(1, r)$ and $\eta = \sqrt{\log(6m/\delta)/(2n)}$. From n gold-audited iid tasks form empirical means of Z_i, U_i and $X_i = rY_i + U_i$. Admit U_i when $\min(1, \widehat{p}_i + \eta) \le \alpha$, admit W_i when $\min(1, \widehat{u}_i + \eta) \le \alpha$, and always admit the exact baseline. Suppose each action's deterministic total price is estimated within ε and selection is within ζ of the smallest estimated admitted loss. With probability at least $1 - \delta$, every admitted action has error at most α, and excess expected loss is at most*

$$2K\eta + 2\varepsilon + \zeta$$

relative to the best action whose true uncertain risk is at most $\alpha - 2\eta$, together with the always-admitted exact baseline.

Proof. Rejection and invalid acceptance are disjoint, so $X_i \in \{0, r, 1\}$ and $0 \le X_i \le K$. Hoeffding's inequality and a union bound over the $3m$ variables $Z_i, U_i, X_i/K$ give simultaneous absolute errors η for the normalized means. Unverified loss error is at most $\eta + \varepsilon$ and checked loss error at most $K\eta + \varepsilon$. Upper risk bounds imply safety. For a comparator with risk at most $\alpha - 2\eta$, its estimated risk plus η is at most α, so it is admitted. Comparing its estimated loss with the selected one and paying the two loss-estimation errors proves the assertion. The exact baseline requires no risk margin. □

Here ε bounds each action's *total additive* deterministic price error, not each constituent price separately. If r is estimated, its contribution must also be covered by a uniform action-total loss-error bound; the displayed proof uses known r. A separate simultaneous interval for q_i requires concentrating $4m$ variables instead, with $\log(8m/\delta)$. The theorem concerns population risk and expected loss, not correctness of each returned witness. Gold calibration costs are paid before deployment. If the checker, population or solver changes, this certificate does not automatically transfer. This is a standard finite-family risk-calibration specialization, closely aligned with Learn then Test's uniform-risk selection result [1, Section 2.5, Proposition 7]; it is not a new general learning principle.

13.1 Variable costs and finite-horizon billing

For task-dependent exact work $R(x)$, expected repair expenditure is $\mathbb{E}[YR]$, not $\mathbb{E}[Y]\mathbb{E}[R]$. On two equiprobable tasks with $R = 10, 1$ and $Y = 1, 0$, the former is 5 and the latter 2.75. End-to-end measured action costs preserve this dependence. With a declared almost-sure bound on complete realized loss, direct bounded-loss calibration is possible; observed wall-clock maxima alone do not supply that bound. A timeout changes the answer interface and must be modelled.

Let C_{A0}, C_{B0} be actual setup costs and $\Delta = L_B - L_A$ the expected deployment saving of a fixed calibrated controller over a specified fixed baseline. Its expected total saving over H further tasks is

$$H\Delta - (C_{A0} - C_{B0}).$$

Only a positive justified lower bound on Δ supports a certified break-even claim. Charging common setup to both sides is appropriate only when both implementations incur it. The study below reports measured bills and empirical projections, not a wall-clock concentration theorem.

14 Public SAT witnesses: calibration, full costs, and negative results

14.1 Frozen primary protocol and probability target

The primary protocol was recorded before downloading or inspecting outcomes. SATLIB's UF20-91 corpus contains 1,000 satisfiable random 3-SAT formulas with 20 variables and 91 clauses [8]. All source files, archive hashes, split, seeds, witnesses and operation timings are retained. A fixed random split assigns 600 formulas to calibration and 400 to testing. Algorithm seeds depend on formula identity, solver and replication, independently of the split. No timeout changes binary outcomes. This supports a fixed finite-population probability statement, not an iid claim about future SAT workloads.

Proposition 14.1 (Held-out finite-population risk certificate)**.** *Let N, n, M be integers with $0 < n < N$, $M \geq 1$, and $0 < \delta < 1$. Fix N binary potential outcomes for each of M predeclared seeded policies. Sample n calibration tasks uniformly without replacement, and observe k errors for a policy. Put $\epsilon = \delta/M$ and define*

$$K^+(k) = \max\{K \in \{k, \ldots, N-n+k\} : \mathbb{P}[\text{Hypergeom}(N, K, n) \leq k] > \epsilon\}.$$

Then simultaneously for all M policies, with probability at least $1-\delta$ over the split, the actual residual-error fraction on the remaining tasks is at most $(K^+(k)-k)/(N-n)$.

Proof. For the true total K, the discrete lower-tail p-value $F_K(X)$ obeys $\mathbb{P}\{F_K(X) \leq \epsilon\} \leq \epsilon$. If K is retained then $K \leq K^+(k)$; exclusion occurs only in that lower-tail event. The feasible set is nonempty since $F_k(k) = 1$. Apply a union bound across policies. Independence between policies is unnecessary. □

This elementary exact inversion certifies the remaining fixed seeded corpus, not fresh seeds, new formulas or drift. It allows data-dependent choice among the predeclared policies but does not transfer the iid theorem's 2η comparator margin or certify timing optimality. An independent integer-arithmetic implementation checks all 2,552 small-population coverage cases and the actual experiment's bounds. For $N = 1000, n = 600, M = 8, \delta = .05$, zero observed errors gives a statistical upper bound $5/400 = .0125$, not zero. Structural zero error under an exactness contract is a separate claim.

14.2 Executed alternatives and accounting

The exact baseline E is a pedagogical Python DPLL solver. Candidate solver S_1 solves the first 45 clauses, filling unspecified variables with false; S_2 performs at most 80 WalkSAT-style flips, with noise probability .5, retaining its already-computed full-formula validity flag. Neither is an industrial approximation baseline. The eight policies are E, unverified U_1, U_2, partial-check-and-repair W_1, W_2, full-check-and-repair F_1, F_2, and trace-aware T_2. The partial checker samples 15 distinct clauses uniformly; T_2 reuses S_2's validity flag and avoids redundant checking. Each path separately executes its actual reads, parsing, solver randomness, checker and conditional fallback. A literal-level full-clause evaluator labels outputs after deployment timing; its calibration work is charged. It shares the parser with the implementation and is not a formal independent parser proof.

Calibration selection minimizes measured seconds plus λ times empirical terminal error for $\lambda \in \{.0001, .001, .01, .1\}$ seconds per invalid answer, subject to held-out risk upper bound .1. Choices are locked before test execution. This is global calibrated selection, not demonstrated per-instance

adaptive control. Primary test replication zero contains 400 tasks; two further seed replications are sensitivity runs on the same tasks, not 800 new independent tasks. Timings are read/parse-inclusive; cold page-cache conditions are not imposed. Raw records provide $.1, 1, 10, 100$-ms exceedance fractions; these are descriptive thresholds, not enforced deadlines.

Policy	Invalid / 400	Mean time (ms)	Calibration error bound
E	0	1.103	.0125
U_1	400	.390	1.0000
U_2	143	1.360	.4050
W_1	149	.975	.4475
W_2	88	1.504	.2650
F_1	0	1.267	.0125
F_2	0	1.726	.0125
T_2	0	1.704	.0125

The last column is the statistical upper bound for the held-out seeded error fraction from calibration, not a confidence interval based on the displayed test errors. Exact and fully repaired policies additionally have structural zero risk under their correctness assumptions. Every constrained choice selects E at all four loss scales. The unconstrained choice selects U_1 only at the smallest scale and E otherwise. Thus this experiment supplies no controller advantage. Recorded setup, calibration and selection consume 7.584055 seconds (including 6.345756 seconds of pipeline work and .157574 seconds of gold labeling). At 400 deployments, fixed E costs about .441 seconds; the learner adds the setup bill and dispatch work. Larger-horizon totals replay empirical means and are explicitly projections. Network transfer, dependency installation and research development are outside this operational setup bill and documented separately.

14.3 Checker calibration and shared blindness

The empirical conditional false-acceptance fractions for S_1 are 221/598 in calibration and 149/400 in the primary test; for S_2 they are 118/196 and 88/143. These ratios are descriptive, not independently certified channel constants. The partial checker cannot reject a genuinely satisfying assignment under the supplied semantics; no valid S_1 candidate occurs in the primary test, so its test conditional false-reject estimate is undefined. The risk certificate instead uses the directly observed joint invalid-acceptance counts.

A separate overlap diagnostic checks the same first 45 clauses that S_1 solved. It accepts all 400 invalid held-out candidates. This is a constructed shared-blindness counterexample, not an estimate for learned verifiers generally. For F_2/T_2, the matched task-level exact-runtime diagnostic gives $\mathbb{E}_n[YR] \approx .3521$ ms against the incorrect product .3228 ms. Actual fallback timings differ slightly because execution is repeated. This illustrates accounting dependence, not a clean causal estimate of physical runtime covariance.

14.4 Exploratory mature-solver and size-transfer amendment

After the primary results, a disclosed amendment adds pycosat 0.6.6's PicoSAT backend [12] and 100 UF50-218 formulas, selected by a fixed filename-hash rule from SATLIB's 1,000-formula corpus. No controller parameters are retuned. PicoSAT is a mature exact baseline, not claimed to be current state of the art. It and a fresh E reference are interleaved on all 400 primary test tasks; all nine policies run on the 100 larger formulas. This amendment is exploratory, and the primary certificate is not transferred.

Corpus	Policy	Invalid / tasks	Mean time (ms)
20 variables	PicoSAT	0 / 400	.282
20 variables	fresh E	0 / 400	1.282
50 variables	PicoSAT	0 / 100	.783
50 variables	E	0 / 100	26.126
50 variables	U_1	100 / 100	.855
50 variables	U_2	87 / 100	5.639
50 variables	W_1	18 / 100	22.670
50 variables	W_2	65 / 100	11.064
50 variables	T_2	0 / 100	28.080

The exact mature baseline is faster and error-free in these recorded comparisons, including against the unverified candidates. Full verification and trace-aware repair do not produce a measured advantage. Partial-checker outcomes change sharply with task size; neither constant channel parameters nor certified drift robustness follow. One machine, small satisfiable random formulas, a limited solver family and no hard deadlines preclude industrial or general efficacy claims. Installation/compiler details, imported binary hash, raw results and selected-file manifest are retained.

15 Contribution boundaries and human-review questions

Learn then Test already permits arbitrary selection from uniformly risk-controlled parameter families and multiple risk constraints [1, Sections 2.4–2.5]. ASlib standardizes algorithm-selection scenarios, instance features and their costs [4, Section 2]; SATzilla already prices portfolio overhead [16]. Recent SHIP work calibrates complete verifier-guided release policies against a separate target judge, with a prespecified grid and explicit calibration expenditure [17, Sections 3.3–3.4, Proposition 2]. Its selective, judge-relative image loss differs from D08's unconditional binary invalid-witness risk after exact repair, but the policy-level calibration principle overlaps directly. Its experiments were not reproduced here. ORACLE already combines model routing, task-dependent verifier choice and asynchronous feedback [13, Sections 4.1–4.2]. Its reported cost excludes classification, verification and routing calls [13, Section 5.1]; D08 prices a different, complete pipeline. This accounting distinction is not a head-to-head performance result or a criticism under ORACLE's declared scope. The joint-event certificate, hypergeometric inversion, conditional-cost identity and setup break-even identity here are applications of established tools. The defensible contribution is their explicit integration with candidate generation, imperfect checking and exact repair, together with preserved negative cases and executable evidence. No claim of being the first such integration is established.

The public-corpus study improves on toy timing evidence by checking 14,800 primary/diagnostic records and adding 1,700 exploratory rows. Independent AI review reconstructs truth labels, seeded checks, exact calibration bounds and billing; six primary formulas also receive complete 2^{20} truth-table enumeration. Finite computation is implementation evidence, not a proof of general solver correctness. All review is simulated/internal AI review; no external human peer review has occurred.

Gold labels resolve rejection-only nonidentifiability in the declared stationary contract, but do not solve calibration without trusted labels, adaptive audit bias, unknown abrupt drift, informative solver traces beyond the stated interface, industrial deadlines or deployment under distribution shift. A useful next study needs a task family where a strong fixed solver is actually expensive, an independently justified checker, a declared loss scale and complete overhead pricing. The concrete human-review questions are whether this limited synthesis merits a methods note, whether the finite-corpus safety target is useful, and whether a proposed next workload can test a plausible

advantage before expensive deployment. Portfolio ownership remains as stated in Section 12; related manuscripts are contextual sources, not shared edited artifacts.

References

[1] Anastasios N. Angelopoulos, Stephen Bates, Emmanuel J. Candès, Michael I. Jordan, and Lihua Lei. Learn then test: Calibrating predictive algorithms to achieve risk control. *The Annals of Applied Statistics*, 19(2), 2025. doi: 10.1214/24-AOAS1998. URL `https://arxiv.org/abs/2110.01052`. Uniform-risk selection: Section 2.5, Proposition 7 in arXiv v5; multiple risks: Section 2.4.

[2] B-Theory research programme. When checking takes time: An exact two-probe stopping frontier, 2026. URL `https://www.b-theory.com/manuscripts-html/when-checking-takes-time/`. Technical addendum, version 1.0, 9 November 2026. No individual author line in supplied source.

[3] B-Theory research programme. Finite noisy audits with learned baselines: Exact risk, interface-dependent memory, and a matched compiled controller, 2026. URL `https://www.b-theory.com/manuscripts-html/finite-noisy-audits/`. Technical addendum, version 1.0, 9 November 2026. No individual author line in supplied source.

[4] Bernd Bischl, Pascal Kerschke, Lars Kotthoff, Marius Lindauer, Yuri Malitsky, Alexandre Frechette, Holger Hoos, Frank Hutter, Kevin Leyton-Brown, Kevin Tierney, and Joaquin Vanschoren. Aslib: A benchmark library for algorithm selection. *Artificial Intelligence*, 237: 41–58, 2016. doi: 10.1016/j.artint.2016.04.003. URL `https://arxiv.org/abs/1506.02465`.

[5] Xueping Gong and Jiheng Zhang. The cost of information: Phase transitions in contextual bandits with paid observations. In *ICML*, volume 306 of *PMLR*, pages 36013–36044, 2026. URL `https://proceedings.mlr.press/v306/gong26i.html`. Primary abstract and metadata checked; full proof audit not claimed.

[6] Nicholas Hay, Stuart Russell, David Tolpin, and Solomon Eyal Shimony. Selecting computations: Theory and applications. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence*, pages 344–353, 2012. URL `https://people.eecs.berkeley.edu/~russell/papers/uai12-meta.pdf`. Proceedings metadata also checked at the PMLR R10 reissue of 4 October 2026.

[7] 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. URL `https://www.cs.rpi.edu/academics/courses/spring06/random/hoefding.pdf`.

[8] Holger H. Hoos and Thomas Stützle. Satlib benchmark problems: Uniform random 3-sat. URL `https://www.cs.ubc.ca/~hoos/SATLIB/benchm.html`. UF20-91 and UF50-218 archives retrieved 13 November 2026; file hashes retained.

[9] Su Jia, Qian Xie, Nathan Kallus, and Peter I. Frazier. Smooth non-stationary bandits. In *ICML*, 2023. URL `https://proceedings.mlr.press/v202/jia23c.html`.

[10] Serdar Kadioglu, Yuri Malitsky, Ashish Sabharwal, Horst Samulowitz, and Meinolf Sellmann. Algorithm selection and scheduling. In *Principles and Practice of Constraint Programming* –

CP 2011, pages 454–469, 2011. doi: 10.1007/978-3-642-23786-7_35. URL https://www.cs.cornell.edu/~sabhar/publications/alg-selection-scheduling-CP2011.pdf.

[11] Yang Liu, Hao Cheng, and Kun Zhang. Identifiability of label noise transition matrix. In *ICML*, 2023. URL https://proceedings.mlr.press/v202/liu23g.html.

[12] pycosat contributors. pycosat: Python bindings to picosat. URL https://github.com/conda/pycosat. Version 0.6.6 pinned for the disclosed exploratory amendment; source and installation receipt retained.

[13] Ritik Raj, Souvik Kundu, Dheemanth Joshi, and Tushar Krishna. Oracle: Agentic ai orchestrator routing via adaptive verifier calibration feedback, 2026. URL https://arxiv.org/abs/2607.22465v4. Version 4, 1 October 2026; Sections 4.1–4.2 and cost scope in Section 5.1 inspected; experiments not reproduced.

[14] Yevgeny Seldin, Peter Bartlett, Koby Crammer, and Yasin Abbasi-Yadkori. Prediction with limited advice and multiarmed bandits with paid observations. In *ICML*, 2014. URL https://proceedings.mlr.press/v32/seldin14.html.

[15] Aaron D. Tucker, Caleb Biddulph, Claire Wang, and Thorsten Joachims. Bandits with costly reward observations. In *UAI*, volume 216 of *PMLR*, pages 2147–2156, 2023. URL https://proceedings.mlr.press/v216/tucker23a.html.

[16] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Satzilla: Portfolio-based algorithm selection for sat. *Journal of Artificial Intelligence Research*, 32:565–606, 2008. doi: 10.1613/jair.2490. URL https://www.cs.ubc.ca/~kevinlb/papers/SATzilla-full.pdf.

[17] Xuanhua Yin, Shunqi Mao, Wei Guo, Chuanzhi Xu, and Weidong Cai. Calibrate what you ship: Post-selection risk control for verifier-guided text-to-image generation, 2026. URL https://arxiv.org/abs/2608.21748. Version 1, 22 August 2026; Sections 3.3–3.4 and Proposition 2 inspected; experiments not reproduced.