OPERATION KOMOREBI

REALIZATION CALCULUS



RESEARCH LIBRARY

AVOC

Artifact value, guarded operations, and calibrated planning

Visual guide

Komorebi P.

Standardized publication edition / 8 October 2026

REALIZATION CALCULUS

Reading scope

From a concrete artifact choice to calibrated policy issuance
A paced companion to the cumulative AVOC v0.41.0 archive

How to read this guide

This guide follows one question through several different kinds of evidence: should a system inspect a cached artifact, generate an artifact, or stop? A recommendation is useful only if the artifact means what the task requires, its declared cost is worthwhile, the chosen operation is permitted, and its reported outcome is handled correctly.

The first worked example is an actual finite-field affine artifact from the archive. A checked width-one representation lowers its declared execution cost from 36 to 21 per call. Warm inspection costs 133. Cold generation costs 196. With 1,000 subsequent calls, their declared net values are 14,867 and 14,804. These are exact values in the stated cost model, not measured hardware savings.

A second example isolates uncertain quote planning. It uses abstract gross values 200 and 60 and is deliberately separate from the affine artifact. Its purpose is to show how probability bands become Q bounds and change a conservative recommendation. It does not provide an artifact with those gross savings.

The last part examines the observation process that feeds those bands. It distinguishes a complete operation log from a justified sampling law. Two exact counterexamples show why conditioning on completion and ordering reports by their outcomes can destroy a proposed statistical interpretation.

Reading route	Main question
Sections 1–6	What is the artifact, and why is either route worthwhile?
Sections 7–15	How do uncertain observations affect values, choices and budgets?
Sections 16–22	How are selected actions, immutable results and replay handled?
Sections 23–27	Which source law makes the final probability guarantee valid?
Sections 28–31	Where do Bayesian reasoning, rare evidence and external trust enter?

Each page introduces one main relationship. Exact arithmetic is displayed as rational numbers when the denominator matters. Decimal labels in figures are reading aids. Formal source files, bounded computations and physical assumptions have different roles throughout the guide.

Source: `FORMAL_THEORY.md`; `PROOF_BOUNDARY.md`; `AVOC/StagedAffineExamples.lean`; `AVOC/RobustQuoteExamples.lean`.

Contents

1 The first decision: inspect, generate, or stop 4

2 Useful compression keeps the distinctions the task needs 5

3 A certificate binds a proof to its exact subject 6

4 The concrete affine artifact: four coordinates through one 7

5 The arithmetic of amortization 8

6 A small change in success probability can reverse the route 9

7 Evidence must travel through the entire chain 10

8 An exact interval built from binomial cells 11

9 More data can change the recommendation 12

10 A probability width becomes a transition-law error 13

11 Bellman values count the continuation 14

12 Broad certified bounds can make the choice conservative 15

13 The learned band improves the same abstract comparison 16

14 Regret accumulates across a policy, not just one action 17

15 Budget is a conserved reservation ledger 18

16 Planning both routes does not execute both routes 19

17 A checked artifact refines deterministic fields 20

18 Recovery distinguishes intent, commit and acknowledgement 21

19 Pending intent belongs outside the immutable terminal log 22

20 Success, rejection and fault have different sample roles 23

21 Replay contributes neither a new sample nor a new reveal 24

22 The generative machine makes terminal samples actual prefixes 25

23 The conditional source law supplies the count law 26

24 A fixed schedule protects data-dependent selection 27

25 Conditioning on completion changes the experiment 28

26 Honest completion order can bias the first report 29

27 The twenty-attempt example violates nominal calibration 30

28 Bayesian belief and calibration answer different questions 31

29 The same first probability can hide different future behavior 32

30 Rare evidence can amplify a tiny likelihood difference 33

31 What the finished chain establishes, and what comes next 34

1 The first decision: inspect, generate, or stop

A reusable artifact may already exist, but its presence in a cache is not proof of its suitability. Inspection must establish the protected subject and check the relevant witness. A cold route must construct an artifact and subject it to the same required semantics. Either route can fail. Stopping remains a permitted action.

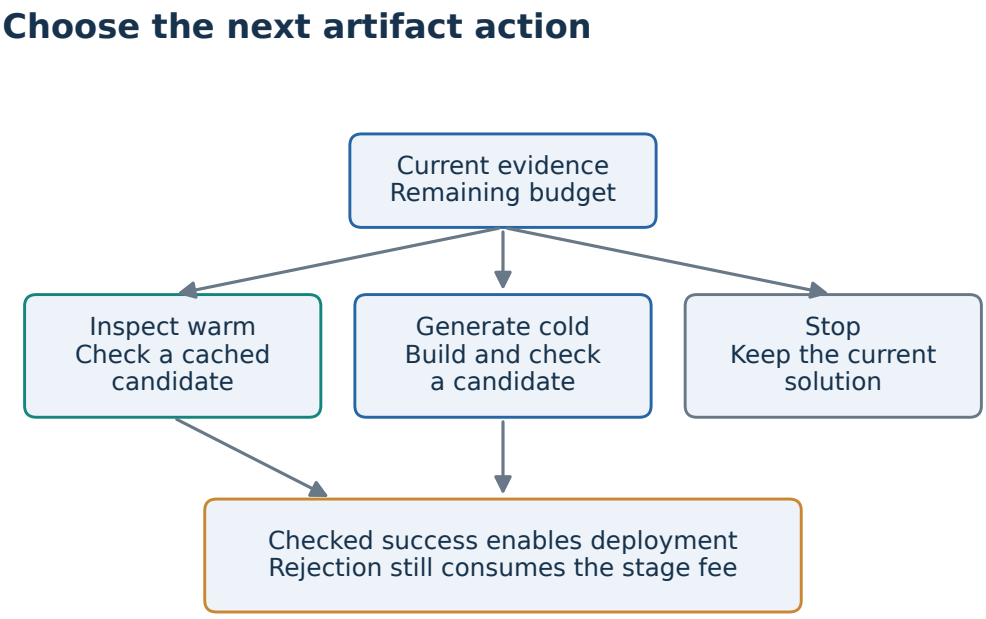


Figure 1: The planning alternatives are quotes for prospective operations. A quote is not an already checked deployment. Only the selected route produces the corresponding actual result. Source: `AVOC/LazyStagedControl.lean` and `AVOC/RobustQuoteRuntime.lean`.

Think of the initial state as *untried*. It contains the task, budget and authorized quote data. It need not contain either completed artifact. Successful inspection moves the warm status to *ready*. Successful generation moves the cold status to *ready*. Deployment becomes available only after the relevant success.

This separation prevents a planner from counting an attractive hypothetical result as work already performed. Planning can compare both routes while the executor performs only the chosen one. A later deployment must use the checked artifact installed by that selected result.

Source: `AVOC/LazyStagedControl.lean`; `AVOC/LazyStagedExamples.lean`; `AVOC/RobustQuoteRuntime.lean`.

2 Useful compression keeps the distinctions the task needs

Suppose the raw state is a pair of bits $x = (a, b)$ and the task is $T(x) = a$. The representation $\phi(x) = a$ merges states with different second bits. Those collisions are harmless because the task gives the same answer on every merged pair. The representation $\psi(x) = b$ merges states with different first bits, so it loses a distinction the task needs.

Raw state	Task T	Useful ϕ	Insufficient ψ	Constant code
$(0,0)$	0	0	0	$*$
$(0,1)$	0	0	1	$*$
$(1,0)$	1	1	0	$*$
$(1,1)$	1	1	1	$*$

The exact criterion is the existence of a decoder d with

$$T = d \circ \phi.$$

With an inhabited target codomain, this is equivalent to requiring that whenever $\phi(x) = \phi(y)$, the target satisfy $T(x) = T(y)$. The bit-valued example meets that condition. The finite table illustrates the archive's general fiber criterion.

A single pair with equal code and unequal target is a complete obstruction to exact recovery. More computation after that code cannot reconstruct a distinction absent from its input. A posterior can express uncertainty about the lost distinction, but it does not turn the code into an exact sufficient representation.

The target determines which collisions matter. If the task changes, a previously useful encoding can become insufficient. If the target is a joint collection of queries, the representation must preserve all distinctions required by that joint target. The archive characterizes this requirement. It does not supply a universal algorithm that finds the cheapest sufficient representation for arbitrary tasks.

Source: `AVOC/Core.lean`: `TaskSufficient`, `collision_obstructs_sufficiency`, `sufficient_iff_fiber_refinement`, `JointTarget`.

3 A certificate binds a proof to its exact subject

A proof receipt has authority only when its protected subject matches the current request and its proof data are checked. A cached Boolean saying “approved” is insufficient. The archive’s propositional backend supports a small data-only proof language with atoms, truth, falsity, conjunction, disjunction and implication.

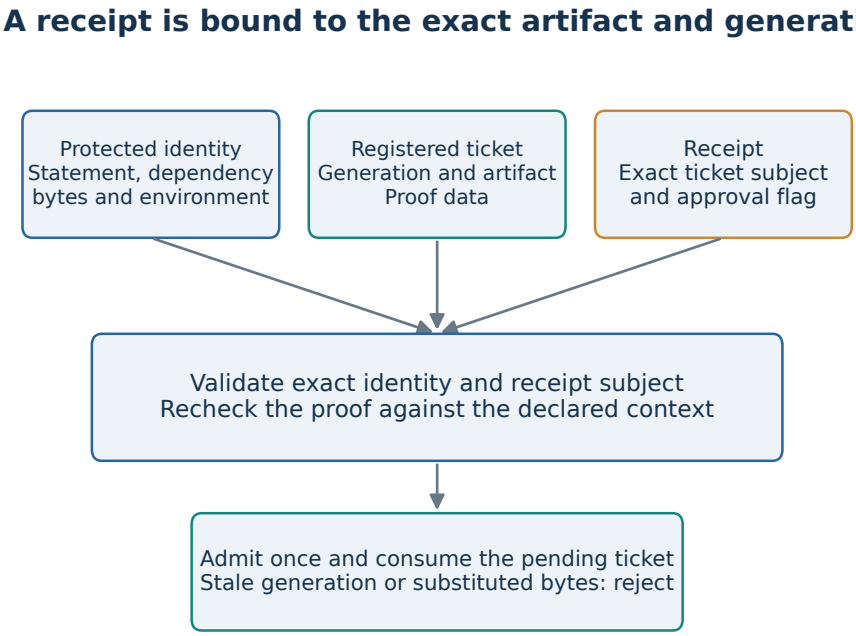


Figure 2: Admission binds the complete protected identity, generation ticket and receipt, then checks the proof. Identity includes exact subject data. No unproved injectivity claim for a hash is used. Source: `ARTIFACT_FORMAT.md`, `AVOC/ArtifactBinding.lean` and `AVOC/CertificateLogic.lean`.

For example, the target $P \to P$ has the proof “assume P, then use that assumption.” In the supported wire format, the target is an implication array and the proof is an introduction containing a hypothesis reference. The soundness theorem concerns the executable inference function on this defined proof syntax.

A probability close to one that a candidate will pass is planning information. It is not a proof of P, and it does not bypass exact admission. Likewise, an identical task name with changed target bytes is a changed subject. Re-registration is intentional.

Source: `ARTIFACT_FORMAT.md`; `AVOC/CertificateLogic.lean`: `check_sound`, `identity_checked`; `AVOC/ArtifactBinding.lean`.

4 The concrete affine artifact: four coordinates through one

The integrated example works over $\mathbb{F}_5$. Its linear matrix A has every entry equal to one and its offset is $b = (1,2,3,4)^{\mathsf{T}}$. Thus the output is determined by one scalar,

$$z = x_1 + x_2 + x_3 + x_4, \qquad y = (z+1, z+2, z+3, z+4)^{\mathsf{T}} \pmod 5.$$

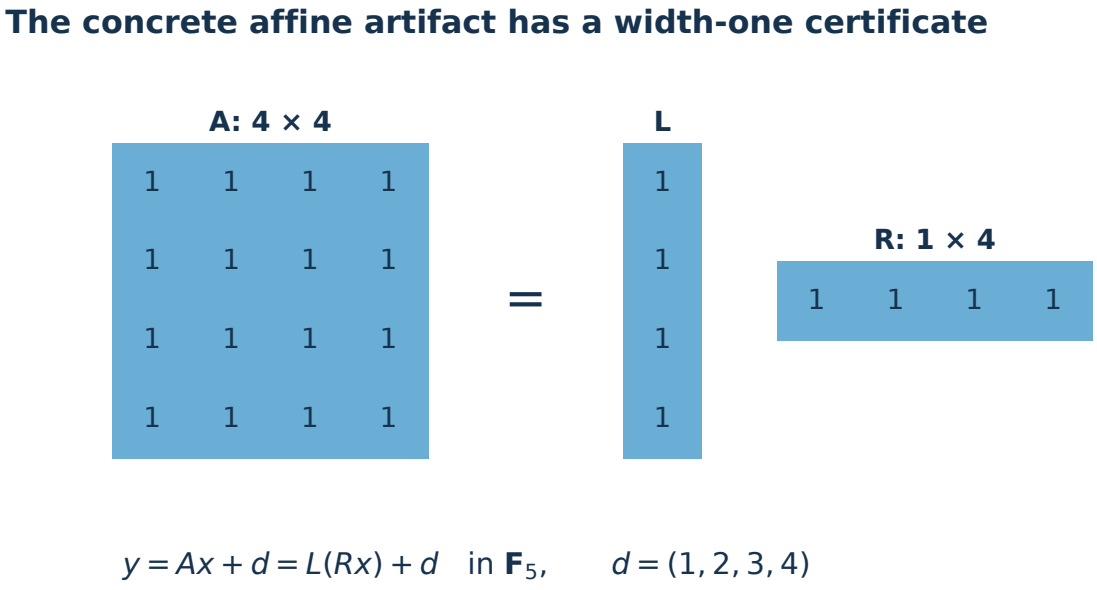


Figure 3: The actual archived map factors as $A = LR$ through a width-one intermediate coordinate. The offset is retained exactly. The figure's exhaustive check covers all $5^4 = 625$ inputs, while the archived Lean theorem establishes the stated factorization and width minimality. Source: `AVOC/AffineFactorExamples.lean`.

The rank witness matters. An exact factorization through two coordinates also exists, but exactness alone would not establish minimum width. The checked candidate includes the identities needed to certify rank one. A candidate with the wrong affine offset is rejected even when its linear product is right.

This is useful compression: the intermediate scalar retains precisely the shared linear quantity needed by all four outputs. Width minimality and execution-cost advantage are different facts. Both must be assessed under their stated languages and cost models.

Source: `AVOC/AffineFactorExamples.lean`: `rankOne_valid`, `rankOne_rank`, `rankOne_minimum_width`, `redundant_lower_rejected`, `wrong_bias_rejected`.

5 The arithmetic of amortization

The source costs 36 declared units per call. The chosen checked implementation costs 21. Therefore the gross saving over N calls is $15N$. Setup must be paid before that saving is credited.

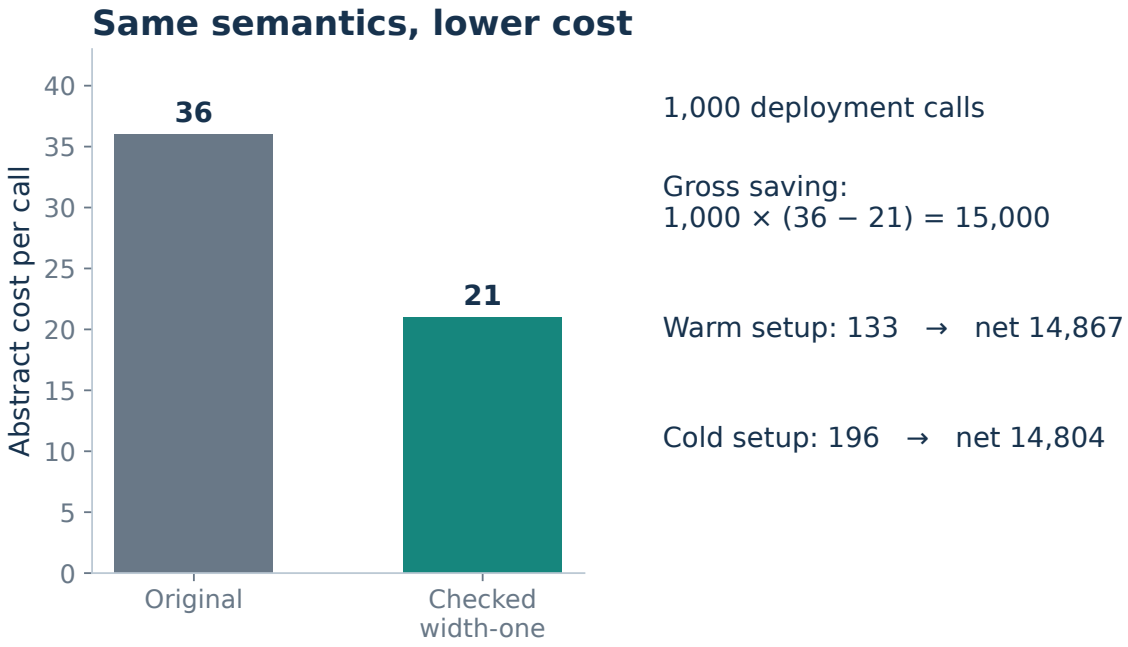


Figure 4: Exact declared cost comparison for the checked rank-one example. Warm setup is 133 and cold setup is 196. At $N = 1{,}000$, the common gross saving is 15,000 and net values are 14,867 and 14,804. Source: `AVOC/StagedAffineExamples.lean`.

The deterministic campaign costs are

$$C_{\text{original}}(N) = 36N, \quad C_{\text{warm}}(N) = 133 + 21N, \quad C_{\text{cold}}(N) = 196 + 21N.$$

Warm is strictly cheaper than the original from $N = 9$, since $15 \cdot 9 = 135 > 133$. Cold is strictly cheaper from $N = 14$, since $15 \cdot 14 = 210 > 196$. At 1,000 calls, warm beats cold by exactly 63 declared units.

These conclusions refer to the declared model. They do not establish elapsed-time savings on a processor, the frequency with which a cache contains a valid artifact, or the cost of collecting success data. The theorem gives a clear accounting target for a later empirical study.

Source: `AVOC/AffineFactorExamples.lean`: `rankOne_execution_costs`, `rankOne_amortization`; `AVOC/StagedAffineExamples.lean`: `gross_saving`, `staged_net_values`.

6 A small change in success probability can reverse the route

Let cold generation succeed with probability one. Let warm inspection succeed with probability q. In the same declared 1,000-call example, a stage-then-deploy warm route has expected net value $15{,}000q - 133$, and the cold route has value $14{,}804$.

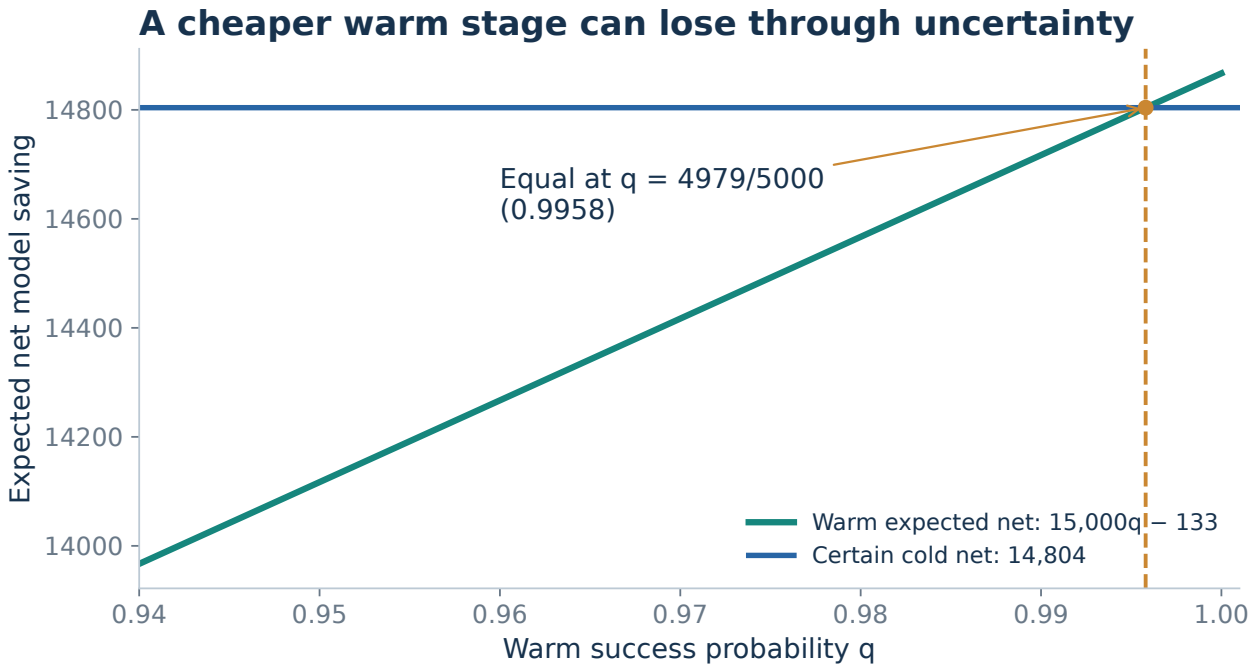


Figure 5: The two route values cross at $q = 4979/5000 = 0.9958$. This exact threshold is a two-action calculation from the archived setup and gross values. It is not an empirical estimate of cache reliability. Source: `AVOC/StagedAffineExamples.lean`.

Solve the comparison:

$$15{,}000q - 133 \geq 14{,}804 \quad \Longleftrightarrow \quad q \geq \frac{4979}{5000}.$$

At $q = 1$, warm's lower setup cost wins. At $q = 1/2$, warm's expected net is 7,367, so certain cold generation is better for this route comparison.

The expected value includes the unsuccessful route: setup is paid even when the checked artifact is not obtained. No successful-branch-only accounting is allowed. A finite-horizon planner can consider additional legal continuations after a failure if the state and remaining budget permit them. The plotted comparison isolates the specified stage-then-deploy routes.

Source: `AVOC/StagedAffineExamples.lean`: `warmCertain`, `warmHalf`, `coldCertain`, `staged_net_values`; `AVOC/LazyStagedControl.lean`.

7 Evidence must travel through the entire chain

The prospective planner needs three kinds of bands: setup cost, gross value and success probability. The artifact backend and authority supply the deterministic setup/gross memberships. The calibration block learns probability bands under a source law and a fixed schedule. These inputs jointly support the policy guarantee.

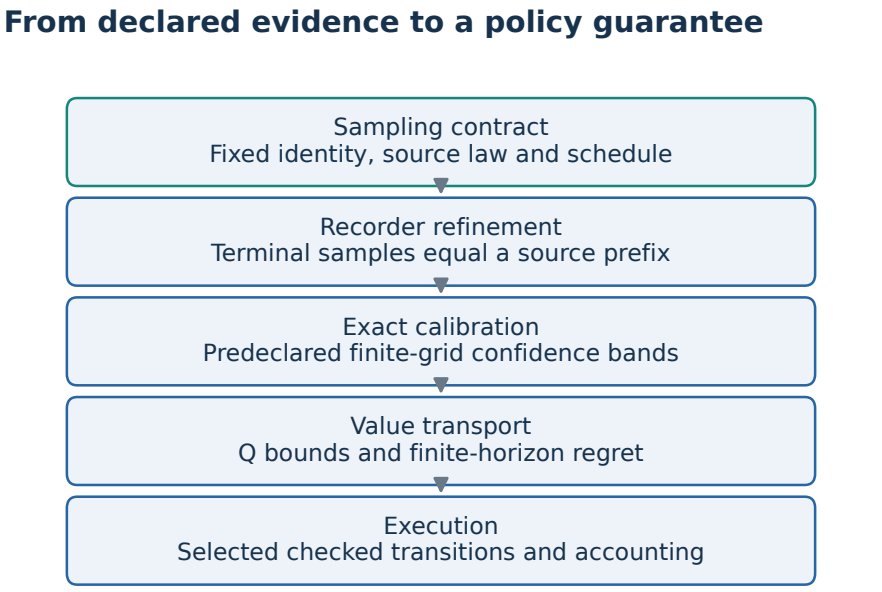


Figure 6: An observation is not a recommendation by itself. Exact samples feed an authorized probability band, which joins deterministic quote memberships, Q bounds and a permitted policy. Source: `AVOC/CalibrationBridge.lean`, `AVOC/CalibratedPolicy.lean` and `AVOC/RecorderPolicy.lean`.

The chain can fail at several distinct points. A candidate can fail proof admission. A log can fail its identity or replay rules. A chosen checkpoint can lack enough reports. A valid interval can still concern the wrong identity-specific probability. A good quote can describe an operation unaffordable in the current state.

The final theorem addresses the composition of the defined steps. It keeps the primitive statistical source assumptions and deterministic memberships explicit. It does not infer those assumptions from the fact that JSON parses or a file exists.

Read each arrow as an obligation with a particular domain. This makes the guide's later counterexamples easy to localize: completion-order bias breaks a proposed physical source interpretation, rather than the arithmetic that turns a valid band into a Q enclosure.

Source: `AVOC/CalibrationBridge.lean`; `AVOC/CalibratedPolicy.lean`; `AVOC/RecorderPolicy.lean`; `PROOF_BOUNDARY.md`.

8 An exact interval built from binomial cells

For a fixed sample size n, let K be the number of successes and assume the specified binomial count law. Its exact count mass at a rational parameter p is

$$f_k(p) = \binom{n}{k} p^k (1-p)^{n-k}.$$

Here $p \in \mathbb{Q} \cap [0,1]$, $m > 0$ and $\alpha \geq 0$. Divide $[0,1]$ into m cells. On cell i, with endpoints $l_i = i/m$ and $u_i = (i+1)/m$, the implementation uses the envelope

$$e_{k,i} = \binom{n}{k} u_i^k (1-l_i)^{n-k}.$$

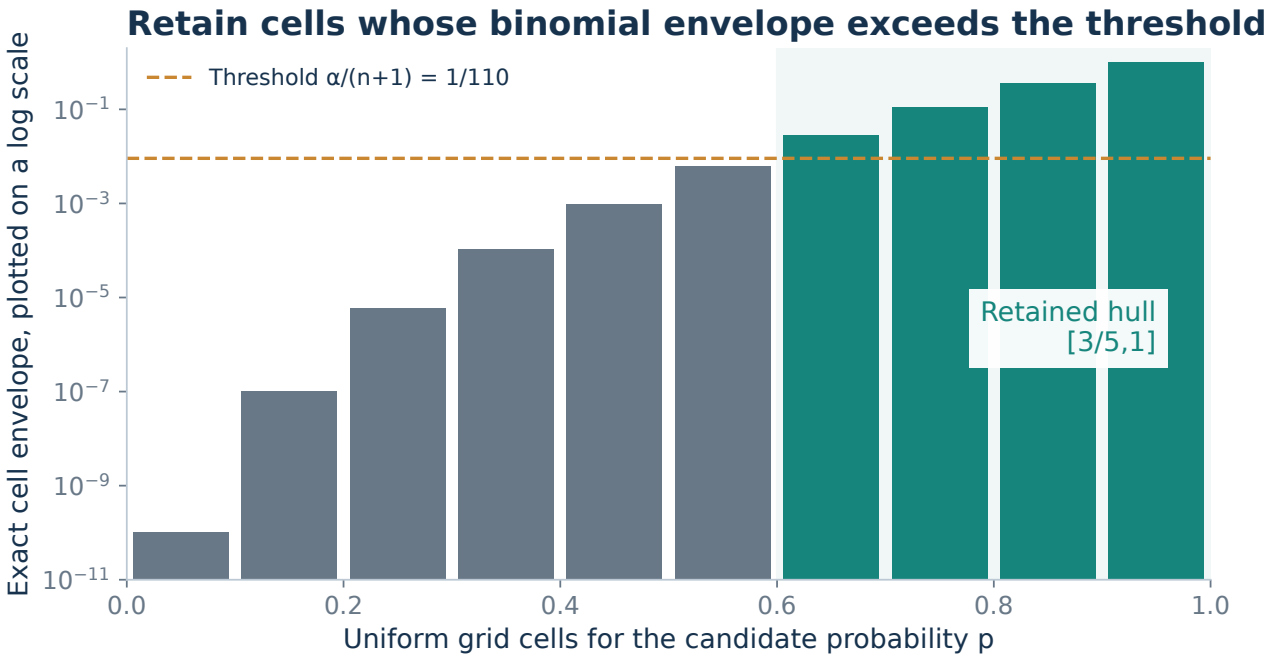


Figure 7: For ten successes in ten trials, ten cells and allowance $1/10$, cells 6–9 are retained. Their hull is $[3/5, 1]$. The true parameter need not lie on a grid point. Source: `AVOC/FiniteCalibration.lean`.

Retain a cell when $e_{k,i} > \alpha/(n+1)$ and return the hull of retained cells. If none are retained, return $[0,1]$. If p is excluded, the observed count mass at that parameter is at most $\alpha/(n+1)$. Sum over the $n+1$ possible counts to obtain miss probability at most α.

This procedure is intentionally simple and conservative. Coverage is proved for the defined interval. Optimal width or computational scalability is not claimed.

Source: `AVOC/FiniteCalibration.lean`: `envelope`, `retained`, `excluded_mass_le`, `missMass_le`, `ten_successes_lower`, `ten_successes_upper`.

9 More data can change the recommendation

The synthetic calibration fixture has 400 successes among 500 checked reports. With allowance $1/100$ and 1,000 prospectively fixed cells, its exact interval is

$$\left[\frac{361}{500}, \frac{433}{500}\right] = [0.722, 0.866].$$

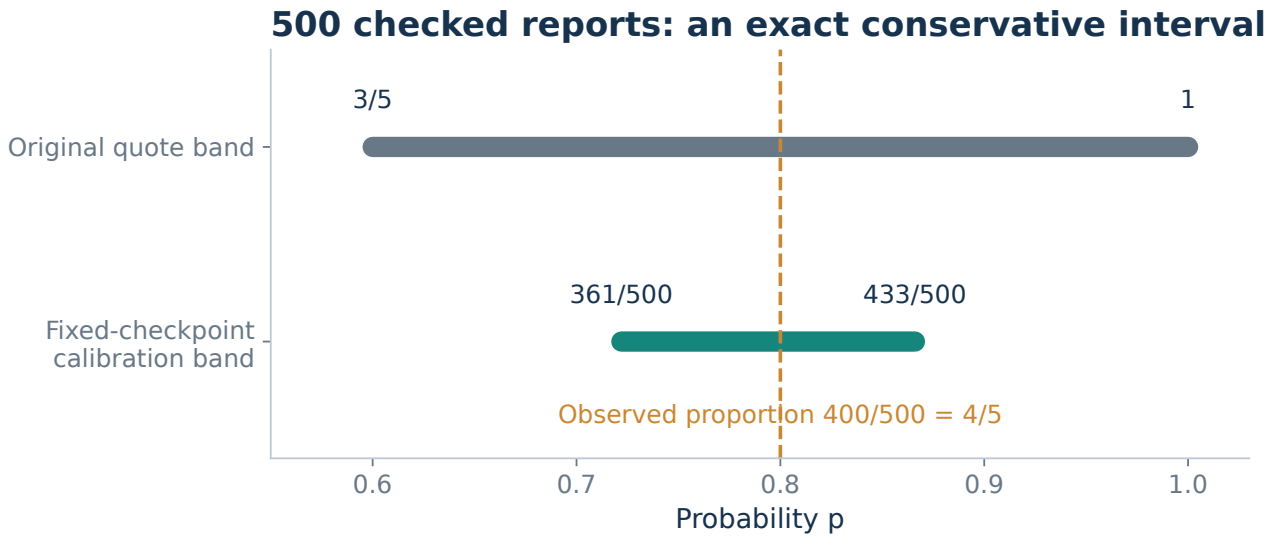


Figure 8: The empirical fraction $4/5$ lies inside the exact returned band. The 100-cell and 1,000-cell authorities are separately fixed scenarios. Choosing a finer grid after observing the result is not part of the claimed prospective protocol. Source: `examples/calibration/README.md`.

The same count with 100 cells gives $[7/10, 22/25]$. The guide later inserts the 1,000-cell band into a separate abstract quote-planning fixture. In that fixture the inherited conservative recommendation changes from cold generation to warm inspection.

The records are synthetic. They are not measurements of an actual provider or cache. The exact numerical reproduction establishes what the implementation returns for those inputs. The universal coverage theorem supplies a different kind of evidence, conditional on its source-law assumptions.

An observed band does not assign posterior probability 0.99 to its particular parameter interval. The coverage statement concerns the frequency of band failure over the specified sampling experiment.

Source: `examples/calibration/README.md`; `examples/calibration/n500_k400_grid1000.json`; `checks/test_calibration.py`; `AVOC/FiniteCalibration.lean`.

10 A probability width becomes a transition-law error

An interval $[l, u]$ authorizes $p \in [l, u]$. The nominal quote uses the lower probability l. A Boolean observation row has probabilities $(p, 1-p)$, so its exact L^1 distance from $(l, 1-l)$ is

$$|p - l| + |(1-p) - (1-l)| = 2(p-l) \le 2(u-l).$$

The factor two is important. It counts both entries of the normalized observation row.

If a continuation function W satisfies $|W| \le B$, the error caused by that row difference is bounded by

$$|\mathbb{E}_p W - \mathbb{E}_l W| \le 2(u-l)B.$$

This is how uncertainty about a prospective success event enters a value calculation. It does not require that an artifact already be present.

Primitive evidence	Planning consequence
Setup width	Immediate stage-reward error
Gross width	Deployment-reward error
Probability width $u - l$	Boolean-row error at most $2(u-l)$
Continuation magnitude B	Row error contributes at most $2(u-l)B$

The archive uses a deliberately simple uniform reward envelope. A wide envelope can produce a broad Q bound even when the true action values have a clear order. Validity and sharpness must be distinguished. Narrower bands improve the bound; a sharper envelope could also improve it, but changing the theorem's implementation requires its own verification.

Source: `AVOC/RobustQuotePlanning.lean`: `bernoulli_l1`, `probability_l1_le`, `expectation_transport`, `rewardEnvelope`.

11 Bellman values count the continuation

For a legal action a in state s, the finite-horizon action value is

$$Q_h(s,a) = r(s,a) + \sum_o P(o \mid s,a) V_h(\mathrm{step}(s,a,o)), \qquad V_{h+1}(s) = \max_{a \in A(s)} Q_h(s,a).$$

Here h is the number of actions in the continuation. Thus Q_1 contains two total actions: the selected action now, and one action afterward.

For an initially untried artifact, a one-action horizon can only pay for inspection or generation. It cannot also deploy the resulting artifact. A longer horizon can recognize the value of successful preparation followed by deployment. In the propositional workflow example, three actions register, verify and admit. Its archived three-step value is $3/2$, while the initial one-step value is zero.

The admissible action set matters as much as the reward. Deployment before a checked success is unavailable. An operation whose upper setup reservation exceeds credit is unavailable. Stopping makes the legal action set nonempty.

The planner itself can also require a work reservation. The archived executable workflow has a depth-three work bound of 1,461 declared planner units. Credit 1,460 rejects that reserved search. This is a bound for the defined evaluator, not a universal statement about the complexity of reasoning.

Horizon	What a preparation workflow can include
1	A selected preparation action only
2	Preparation, then deployment on success
3	Register, verify, then admit in the propositional example

Source: `AVOC/GuardedControl.lean`; `AVOC/ArtifactWorkflowExamples.lean`; `AVOC/PlannerExamples.lean`: `executable_three_step_value`, `myopic_value_zero`, `depth_three_reservation`.

12 Broad certified bounds can make the choice conservative

This page switches to an *abstract quote-planning fixture*. Both routes start untried, setup is zero, and $h = 1$ means two total actions. Warm gross is 200 with probability band $[3/5, 1]$. Cold gross is 60 with exact probability one. These gross values are not achievable savings for the separate one-call rank-one artifact, whose source cost is only 36.

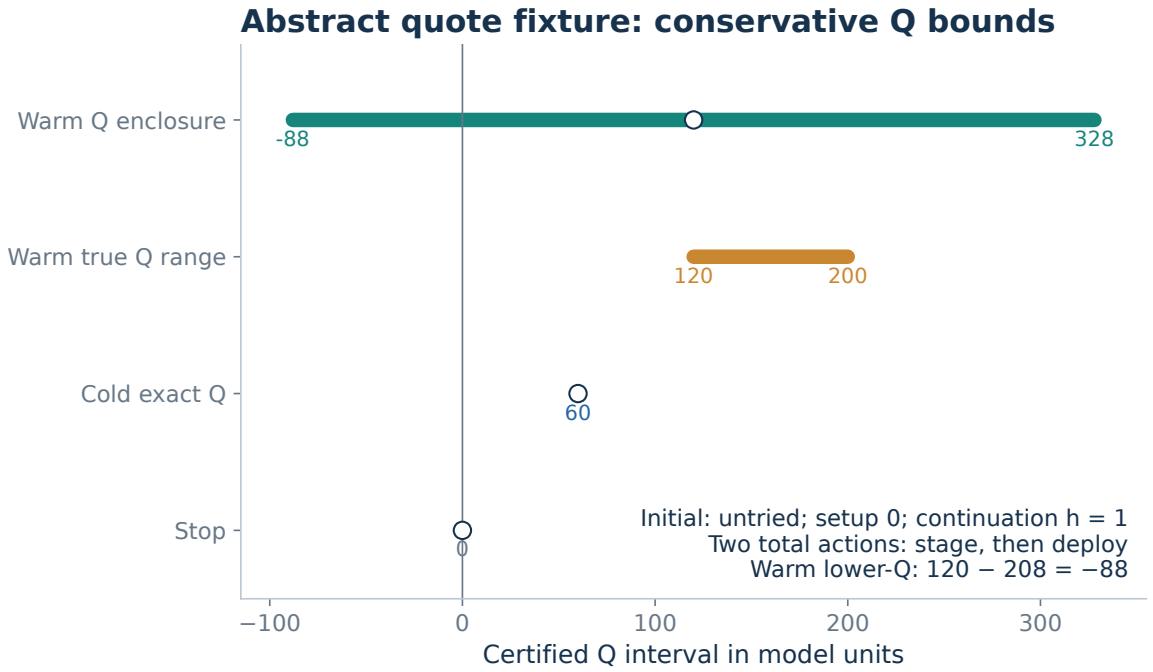


Figure 9: Warm nominal Q is 120 and its inherited certified radius is 208, giving $[-88, 328]$. Cold Q is exactly 60. The direct true warm stage-then-deploy Q range is $[120, 200]$, which exposes how broad the certified bound is. Source: `AVOC/RobustQuoteExamples.lean`.

The conservative selector maximizes certified lower Q. It chooses cold because $60 > -88$ and $60 > 0$, where zero is the stop score in this state and horizon. This does not certify cold as the actual best action. Indeed the fixture's direct two-action warm value is at least 120.

The example demonstrates valid conservativeness, not an optimality advantage. A Q enclosure can support a regret bound while being too broad to preserve a clear direct value ordering.

Source: `AVOC/RobustQuoteExamples.lean`: `uncertain_probability_width`, `uncertain_warm_local_radius`, `uncertain_cold_local_radius`, `uncertainty_changes_numerical_order`; `examples/robust_quote/uncertain_plan.json`.

13 The learned band improves the same abstract comparison

Keep the abstract setup-zero, initially untried, two-action fixture, and replace the warm probability band before a fresh episode by $[361/500, 433/500]$. Its width is $18/125$. The nominal warm Q and radius become

$$\widehat{Q}_1(\text{warm}) = \frac{722}{5}, \qquad \rho_1(\text{warm}) = \frac{1872}{25}.$$

Therefore its lower Q is $1738/25 = 69.52$, above the exact cold value 60.

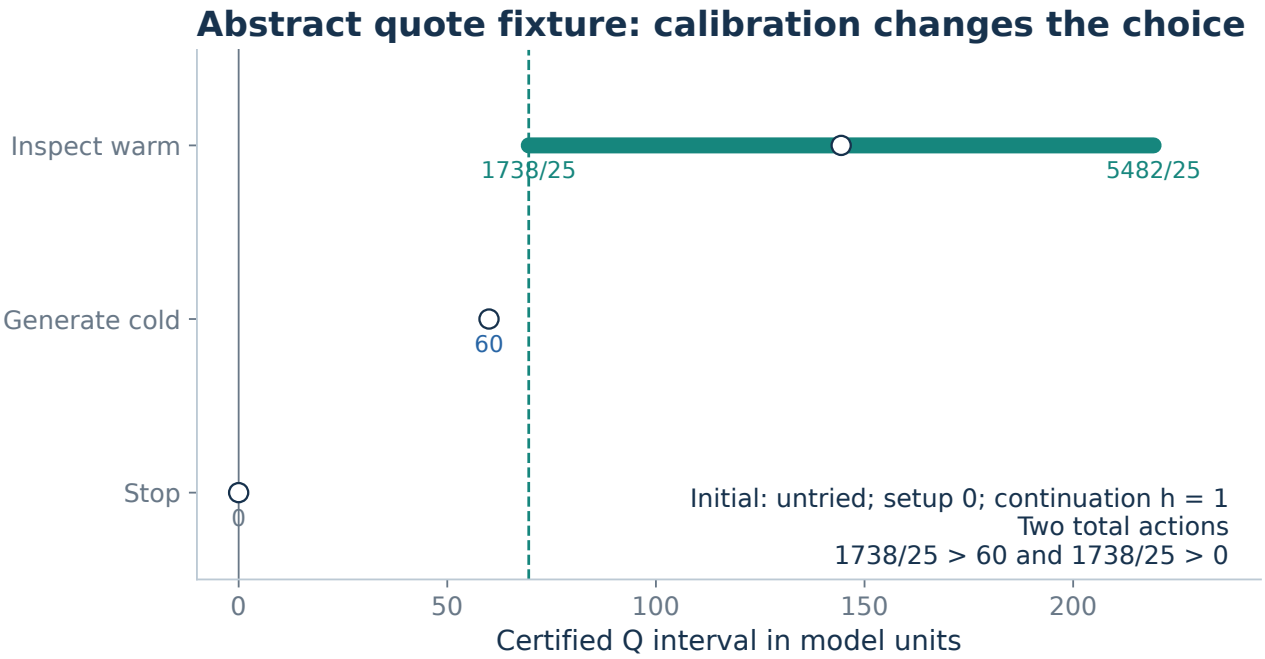


Figure 10: Warm's certified lower Q exceeds the upper Q of each distinct feasible competitor: cold at 60 and stop at zero. This certifies the two-action warm choice within the abstract quote fixture, on valid membership in the learned band. Source: `AVOC/CalibrationBridge.lean` and the archived n500 fixture.

The corrected dominance test compares $\underline{Q}(a)$ with $\overline{Q}(b)$ for $b \neq a$. Comparing an action with its own upper Q would require $\overline{Q}(a) \leq \underline{Q}(a)$, forcing zero radius. That restrictive self-comparison is sound, but prevents useful certificates for an uncertain selected action.

The learned band improves the decision while the statistical source assumptions remain unchanged. It does not authenticate those assumptions. The gross values remain an abstract planning illustration, not a deployable affine savings claim.

Source: `AVOC/CalibrationBridge.lean`: `CompetitorDominance`, `competitor_dominance_sound`, `self_comparison_forces_zero_radius`; `examples/calibration/README.md`; `results/v0.39.0/calibration_lean_replan_n500_grid1000.json`.

14 Regret accumulates across a policy, not just one action

Let $\rho_h(a)$ enclose the local Q error and let R_h be a uniform bound on those radii. Choosing the greatest certified lower Q gives a local action gap at most $2R_h$. The factor two accounts for uncertainty in the best comparator and in the selected action.

The archive carries that local bound through the entire adaptive finite-horizon policy. In simplified notation,

$$0 \le V_H^{\text{true}}(s) - V_H^{\text{policy}}(s) \le \sum_{h=0}^{H-1} 2R_h.$$

The left side is an expected policy-value difference. It is not the loss on every realized trajectory. A low-probability unsuccessful branch can have a poor realized outcome while the expected guarantee holds.

The comparator uses the same feasible action guards as the nominal planner, including upper setup reservation. It is not an omniscient policy allowed to spend each route's smaller realized setup cost and thereby unlock actions the planner cannot use.

Statement	Meaning
Q enclosure	Each compatible realized Q lies between the displayed bounds
Local action gap	Selected action is within a certified expected gap at that state
Whole-policy regret	Expected loss of the induced finite policy is bounded cumulatively
Sample-path loss	A different quantity, not bounded by this theorem

Exact singleton bands make all quote-induced radii vanish. They recover the inherited exact selector. This limiting check explains how uncertainty extends the exact model without changing its established deterministic case.

Source: `AVOC/RobustQuotePlanning.lean`: `robust_action_gap`, `robust_policy_regret`, `exact_localQRadius`, `robustChoice_exact`; `AVOC/GuardedPolicyRegret.lean`.

15 Budget is a conserved reservation ledger

Return to the artifact-backed rank-one example with initial credit 196. Warm inspection reserves 133, leaving 63. Cold generation reserves 196, leaving zero. The formal invariant is

$$\text{remaining} + \text{spent} = 196.$$

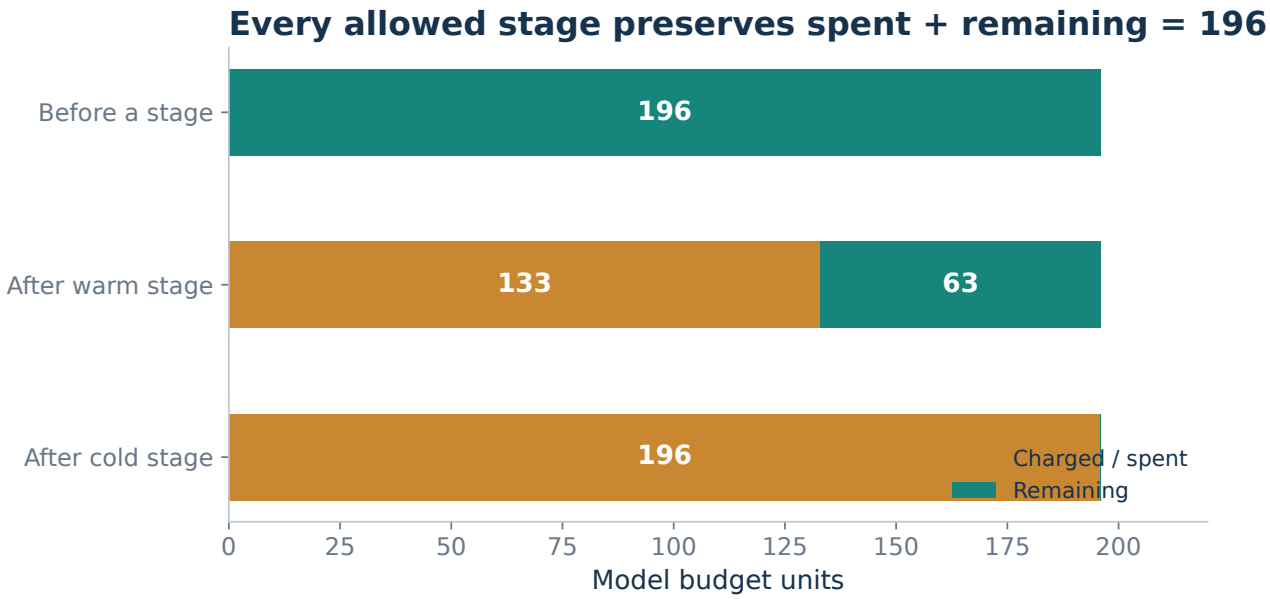


Figure 11: The two routes conserve their initial declared reservation credit. The spent field records authorized setup reservations; it is not an automatically measured physical-spending field. Source: `AVOC/StagedAffineExamples.lean`.

The reservation uses an upper setup bound so an admitted action remains feasible for every compatible setup realization. A smaller actual setup value does not automatically refund the difference. After warm's reservation, cold generation at fee 196 is unavailable with only 63 remaining.

An operation that fails or is interrupted can still have consumed its authorized reservation. Deleting the failure record cannot erase that cost. Repeating an acknowledgement of the same committed result adds no new nominal debit, while authorizing genuinely fresh paid work needs a fresh operation identity and sufficient credit.

The budget theorem is precise because it names what the ledger records. Calling the ledger a physical cost meter would add an unsupported interpretation.

Source: `AVOC/StagedAffineExamples.lean`: `staged_budget_after_warm`, `staged_budget_after_cold`; `AVOC/RobustQuoteRuntime.lean`: `confirm_budgetSafe`; `RECOVERY_FORMAT.md`.

16 Planning both routes does not execute both routes

A planner compares warm and cold quotes. That comparison is a pure calculation over the authorized model. It does not need to construct both actual artifacts in advance. The source example explicitly proves that its recommendation is unchanged when artifact slots are prefilled: recommendation depends on the plan state and quotes.

The selected executor then obtains the chosen operation's checked result. Warm success installs the warm artifact and its refined evidence. The cold artifact and cold evidence remain unchanged. Cold success has the symmetric behavior. A model Boolean without the required checked object is not enough to install a successful deployment.

Phase	What it establishes
Quote comparison	A permitted prospective action under the model
Selected stage	An actual result for that chosen route
Confirmation	Correspondence between the result and the plan transition
Installation	The chosen checked artifact is available for deployment

This makes the execution claim reviewable. One can ask exactly which route was invoked, which artifact was checked, which protected identity it had, which debit was recorded and what remained untouched.

The principle does not claim that speculative computation is always wasteful. It gives the semantics of this selected-stage runtime. Likewise, the new lifecycle recorder introduced later in the guide merely records supplied commands; it is not the executor described on this page.

Source: `AVOC/LazyStagedExamples.lean`: `recommendation_does_not_require_artifacts`; `AVOC/RobustQuoteRuntime.lean`: `warm_success_stores_refinement`, `warm_success_preserves_cold`, `confirm_exact`.

17 A checked artifact refines deterministic fields

Successful artifact admission can establish the actual setup and gross value of that artifact under the declared semantics. It does not establish the success probability of future attempts at the provider.

In the exact rank-one warm example, the installed refined evidence has

$$\text{setup} = 133, \qquad \text{gross} = 15{,}000, \qquad \text{probabilityBand} = \text{the authorized warm band.}$$

The first two fields become exact through the checked deployment. The probability band is retained, because a single artifact is not a statistical proof about the next operation's outcome.

A protected quote identity includes the source, workload calls, kind, provider, checker, cost model, environment and schema. A probability band for one identity cannot be silently attached to another. A band refresh also changes evidence identity, so an old durable acknowledgement cannot be reused as if it belonged to the new evidence.

The policy episode freezes quote data. Learning a new band happens before a fresh episode. The archive does not authorize changing the model beneath an outstanding intent or updating quotes arbitrarily during a running episode.

Checked now	Remains a separate obligation
Artifact semantics	Provider's future statistical source law
Declared deterministic costs	Empirical elapsed-time calibration
Exact protected subject	Authenticity of externally supplied logs
Current plan transition	General cross-episode accounting

Source: `AVOC/RobustQuoteExamples.lean`: `artifact_refines_deterministic_fields`; `AVOC/QuoteAuthority.lean`; `AVOC/CalibrationBridge.lean`: `changed_band_changes_evidence`, `new_bands_obstruct_old_acknowledgement`.

18 Recovery distinguishes intent, commit and acknowledgement

The inherited durable run protocol first records an intent and conservative debit. It then executes the selected stage and publishes a checked result plus resulting state atomically under its stated storage contract. Finally, the monitor acknowledges that durable result.

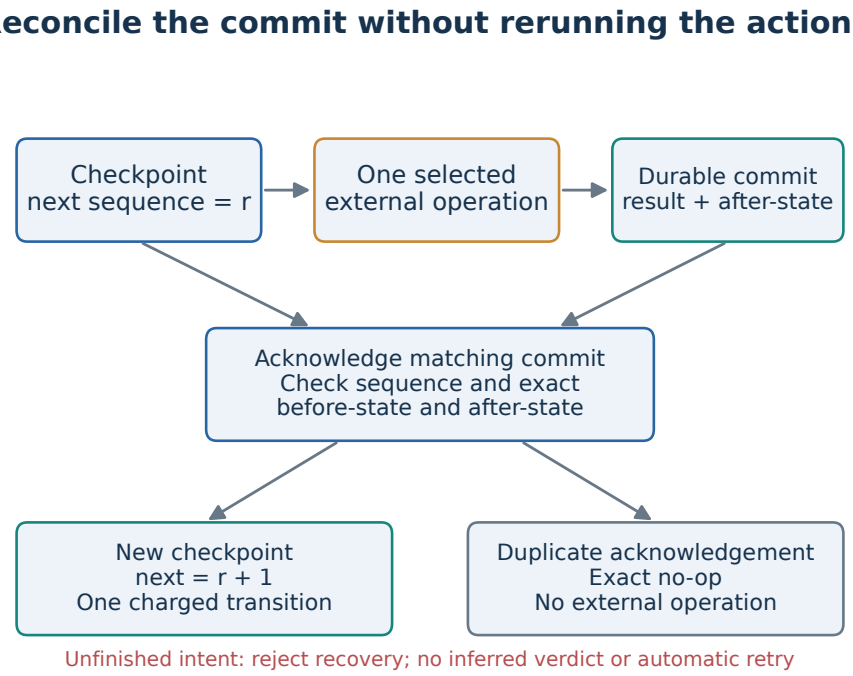


Figure 12: A crash before a durable result leaves an unresolved intent. A crash after result publication can be reconciled without rerunning the selected stage. Duplicate acknowledgement is idempotent. Source: `RECOVERY_FORMAT.md` and `AVOC/RobustQuoteRecovery.lean`.

An unfinished intent is not evidence of success or failure. Recovery must not invent a report or automatically repeat paid work. A committed result can be acknowledged again, because replay concerns the recorded result rather than a fresh external operation.

The concrete warm example proves that its committed result recovers without a rerun and that exact repeated acknowledgement returns the same checkpoint. The surrounding Python/SQLite adapter has tested process-interruption behavior. Its storage guarantees still depend on the protected directory, cooperative coordinator and SQLite/OS transaction contract.

This protocol is distinct from the v0.41 observation recorder, which is a pure snapshot reducer and writes no durable journal itself.

Source: `RECOVERY_FORMAT.md`; `AVOC/LazyStagedExamples.lean`: `unfinished_recovery_is_not_evidence`; `AVOC/RobustQuoteExamples.lean`: `committed_result_recovers_without_rerun`, `duplicate_acknowledgement_is_exactly_idempotent`.

19 Pending intent belongs outside the immutable terminal log

The calibration log is immutable: changing the outcome associated with an existing operation ID is rejected. If an unfinished event were placed into that terminal log, appending a completed result under the same ID would be a conflicting reuse. The v0.41 machine resolves this mismatch by storing pending intent separately.

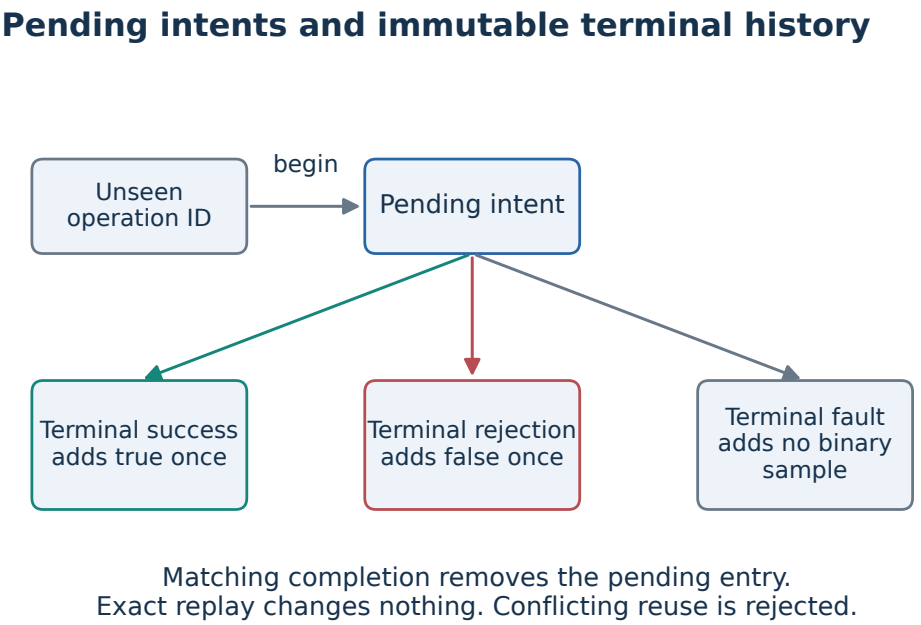


Figure 13: A global operation ID occupies either the pending registry or the immutable terminal history. Fresh completion requires the matching pending identity, clears that entry and appends one terminal event. Source: `AVOC/ObservationRecorder.lean`.

The namespace is global across identities. A warm operation and a cold operation cannot legitimately share an ID merely because they have different identity keys. Exact repeated begin does not add another pending entry. A late begin after matching completion does not reopen the operation.

The well-formedness theorem covers every accepted finite run from the empty state: pending and terminal IDs are unique and disjoint, and terminal history has no unfinished outcomes. An accepted terminal event is retained exactly once through later successful continuations.

These are statements about the defined machine. One terminal record per ID is not proof that the corresponding external work executed exactly once. The same physical work could be assigned a fresh ID unless an additional external protocol rules that out.

Source: `AVOC/ObservationRecorder.lean`: `inherited_unfinished_terminal_conflict`, `run_wellFormed`, `accepted_terminal_retained_once`; `FORMAL_THEORY.md`.

20 Success, rejection and fault have different sample roles

A checked success contributes the Boolean true. A checked rejection contributes false. Both advance the checked-report sample count. A fault closes its pending operation but contributes no binary sample.

Terminal event	Clears intent	Adds sample	Adds success	Immutable
Success	Yes	1	1	Yes
Rejection	Yes	1	0	Yes
Fault	Yes	0	0	Yes

A first-n checkpoint asks for n checked reports, not n successes and not n initiated operations. Faults therefore do not help it become ready. The log also retains other identities without mixing their sample lists.

The model's "fault consumes no latent accepted-source position" rule is a statistical modeling choice, not a synonym for any physical timeout. If a real system first observes a rejection and then relabels it as a fault, it may be discarding an unfavorable checked report. Such consuming censoring can change the target probability.

Raw commands supply terminal labels. The source-generative machine introduced next constructs labels from latent bits. An exporter or parser can enforce the former's syntax and identity rules without proving the latter's probability law.

This distinction prevents two errors: treating rejected checks as absent observations, and treating infrastructure labels as automatically independent of the outcome being measured.

Source: `AVOC/ObservationRecorder.lean`: `fresh_success_observed_once`, `fresh_rejection_observed_once`, `fault_preserves_samples`; `OBSERVATION_EXECUTABLE.md`; `PROOF_BOUNDARY.md`.

21 Replay contributes neither a new sample nor a new reveal

Exact terminal replay finds the existing event and returns the same state. A changed identity or changed outcome under the same ID is rejected. Replay is handled before asking the source for a fresh outcome.

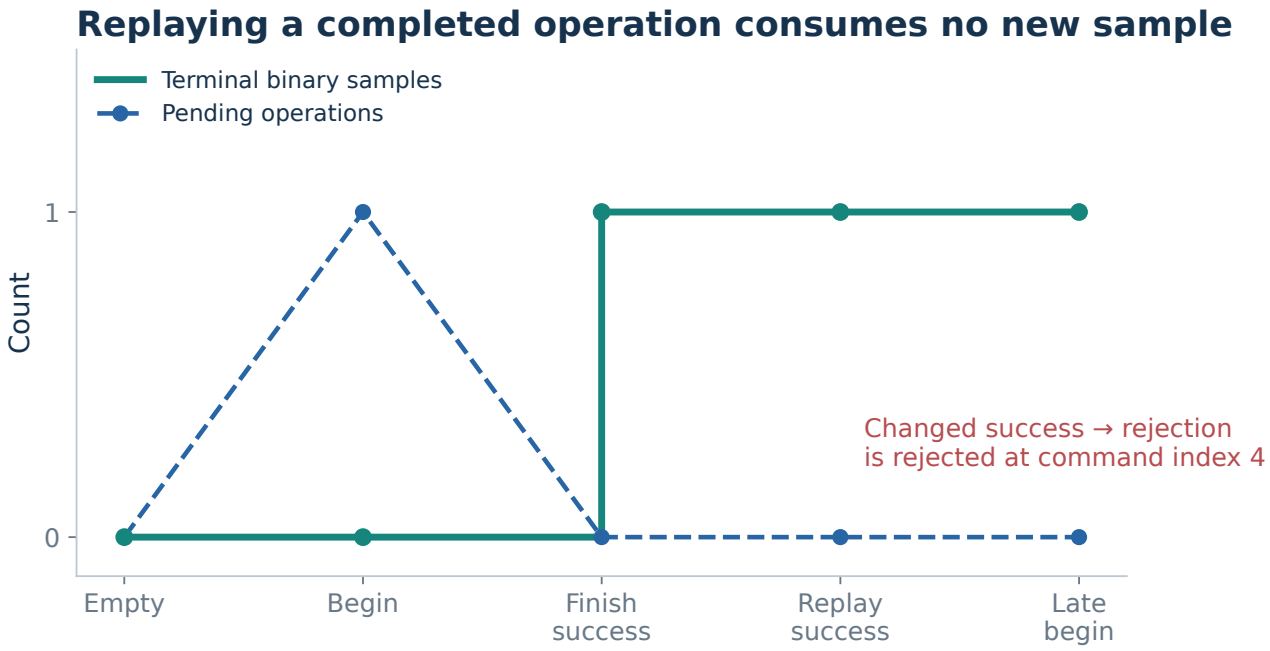


Figure 14: The first completion appends one event. Replaying the identical terminal result leaves the event count and sample count unchanged. Conflicting reuse fails. Source: `AVOC/ObservationRecorder.lean` and `AVOC/RecorderReveal.lean`.

This ordering matters when a latent source is exhausted. Replaying a previously completed report still succeeds, because it needs no next bit. A genuinely fresh report from an exhausted source is rejected. A report after a fault conflicts with immutable terminal history; fault after an already reported result does likewise.

Replay idempotence is a state-machine guarantee. Receiving the same network request twice does not become proof that an external paid action was never repeated. The pure reducer operates on commands already supplied to it. External execution and durable recovery require their own contracts.

Likewise, failed batch reduction yields no final snapshot. It does not roll back actions performed outside the reducer before their commands were submitted.

Source: `AVOC/ObservationRecorder.lean`: `exact_terminal_replay`, `changed_terminal_rejected`; `AVOC/RecorderReveal.lean`: `replay_fault_exhaustion_example`, `exhausted_fresh_report_rejected`; `RECORDER_EXECUTABLE.md`.

22 The generative machine makes terminal samples actual prefixes

For each protected identity k, fix a finite latent Boolean list S_k. A fresh report consumes its next bit and constructs the corresponding terminal success or rejection. A replay consumes none. A fault contributes no accepted-source position.

The machine retains the pathwise identity

$$\mathrm{samples}(k, \mathrm{terminalLog}) = S_k.\,\mathrm{take}(|\,\mathrm{samples}(k, \mathrm{terminalLog})|)\,.$$

This is a statement about actual terminal samples of the defined generated run. It is derived by refinement, rather than separately assumed for every fresh raw-log event.

For example, if $S_k = [\mathrm{true}, \mathrm{false}, \mathrm{true}]$, begin/report for the first ID appends true. A fault on a second ID leaves the next report bit false. A third fresh report appends false. Replaying the first report changes neither the list nor the next position.

The command trace can depend on outcomes. The prefix invariant still holds, because the transitions never skip the next latent report. Adaptive allocation among identities does not require independently randomized scheduling for this pathwise result.

This theorem does not test whether an arbitrary provider's reported bit equals the intended latent bit. A raw endpoint accepts supplied labels, whereas the generative transition constructs them. Applying the final probability theorem to physical inputs requires a justified bridge between those situations.

Source: `AVOC/RecorderReveal.lean`: `run_refines`, `run_samples_eq_take`, `completed_samples_eq_source`; `AVOC/ObservationReveal.lean`; `FORMAL_THEORY.md`.

23 The conditional source law supplies the count law

The statistical model works on a finite rational probability space. For identity k, each source position has a fixed rational success probability p_k. The primitive condition says that specifying outcomes at other positions leaves an unseen position's success mass equal to p_k times the mass of that specified pattern.

This is expressed as a multiplication identity. Impossible histories have zero mass and require no undefined division. Induction over positions derives the probability of each complete vector:

$$p_k^{\#\text{true}}(1-p_k)^{\#\text{false}}.$$

Counting vectors with s successes gives the binomial count mass $\binom{n}{s}p_k^s(1-p_k)^{n-s}$. Restricting to a prefix preserves the same contract.

Equal one-position marginals are insufficient. Two perfectly correlated fair bits have outcomes 00 and 11, each with mass $1/2$. Both bits are fair, yet their success count never equals one. A binomial two-trial count would equal one with probability $1/2$.

Multiple checkpoints for an identity can share the same source. Their counts need not be independent. Cross-identity independence is also unnecessary for the final scheduled union bound. The crucial independence requirement is within each specified source, in the precise conditional-law form used by the theorem.

The theorem derives count laws from this primitive contract. It does not derive the contract from log completeness, parser acceptance, matching operation IDs or the user-visible fact that attempts looked unrelated.

Source: `AVOC/ObservationStream.lean`: `conditional_pattern_mass`, `conditional_implies_product`, `prefix_count_fiber_eq_binomial`; `provenance/v0.40_release_top/FORMAL_THEORY.md`.

24 A fixed schedule protects data-dependent selection

Before the relevant data are observed, checkpoint j fixes its identity, sample size n_j, positive grid size m_j and nonnegative allowance α_j. Let B_j be the event that this checkpoint's interval excludes its true target parameter. Then

$$\mathbb{P}(B_j) \le \alpha_j, \qquad \mathbb{P}\left(\bigcup_j B_j\right) \le \sum_j \alpha_j.$$

The union bound does not require independent checkpoints.

Selectors can choose warm and cold checkpoints after seeing outcomes, provided they select from this predetermined schedule and match the specified target strata. Any selected bad interval is contained in the already budgeted union. Selection does not create an unbudgeted new checkpoint.

The recorder's gates add three concrete requirements. The full generated run must be accepted. Every pending operation must be resolved, including unrelated identities. Each selected checkpoint must have enough checked reports. Otherwise there are no counts for issuance and no plan.

The final conclusion is

$$\boxed{\mathbb{P}(\text{issued plan and not EpisodeCertified}) \le \sum_j \alpha_j.}$$

All rejected, unresolved and insufficient-sample runs remain in the original probability space. They issue no plan. This makes the conclusion unconditional over those runs.

The bound does not guarantee a high issuance rate. A process that never issues a plan has zero bad-issuance probability. Availability, collection cost and useful throughput need additional results.

Source: `AVOC/CalibrationSchedule.lean`; `AVOC/ObservedCalibration.lean`; `AVOC/RecorderPolicy.lean`: `pending_run_no_plan`, `recorder_issued_failure_le`.

25 Conditioning on completion changes the experiment

Consider two fair latent bits. Always report the first bit. Report the second only if the first was a success. The decision to continue does not inspect the second bit. Nevertheless, conditioning on completion means conditioning on the first bit being true.

Predictable availability does not justify conditioning on completion

First bit	Second bit	Complete?
R	R	No
R	S	No
S	R	Yes
S	S	Yes

Report the second bit only if the first was successful.
Every pair has mass 1/4. Completion has mass 1/2.

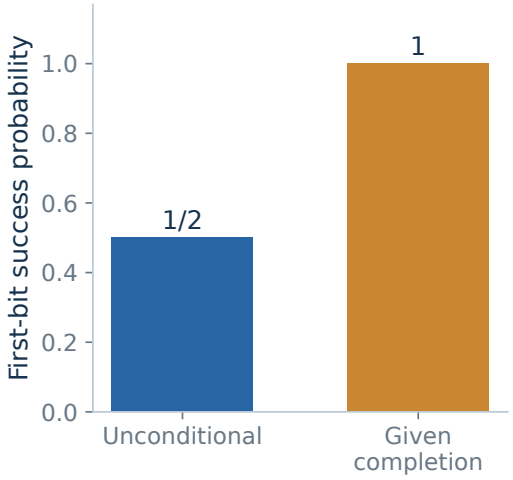


Figure 15: Completed runs have success-count probabilities $(0, 1/2, 1/2)$, rather than the unconditional binomial $(1/4, 1/2, 1/4)$. Completion probability is $1/2$. Source: `AVOC/ObservationCensoring.lean`.

The final safe theorem bounds $\mathbb{P}(\text{issue} \cap \text{bad})$. It does not automatically bound $\mathbb{P}(\text{bad} \mid \text{issue})$ by the same allowance. Division by the issuance probability can enlarge the bound.

The archived interval-specific exact experiment makes this visible with ten fair latent bits. Permit the tenth report only after nine successes. At grid ten and allowance $1/10$, conditional miscoverage is $1/2$, while unconditional bad issuance is $1/1024$.

The first obstruction is kernel checked. The larger interval-specific calculation is exact finite executable evidence. Both explain why a nominal guarantee conditioned on completion is outside the safe theorem's conclusion.

Source: `AVOC/ObservationCensoring.lean`; `provenance/v0.40_release_top/FORMAL_THEORY.md`; `OBSERVATION_EXECUTABLE.md`.

26 Honest completion order can bias the first report

Now initiate two independent fair attempts and retain both outcomes. Let a successful attempt finish before a rejecting attempt. Every intent is resolved and every result is logged exactly once.

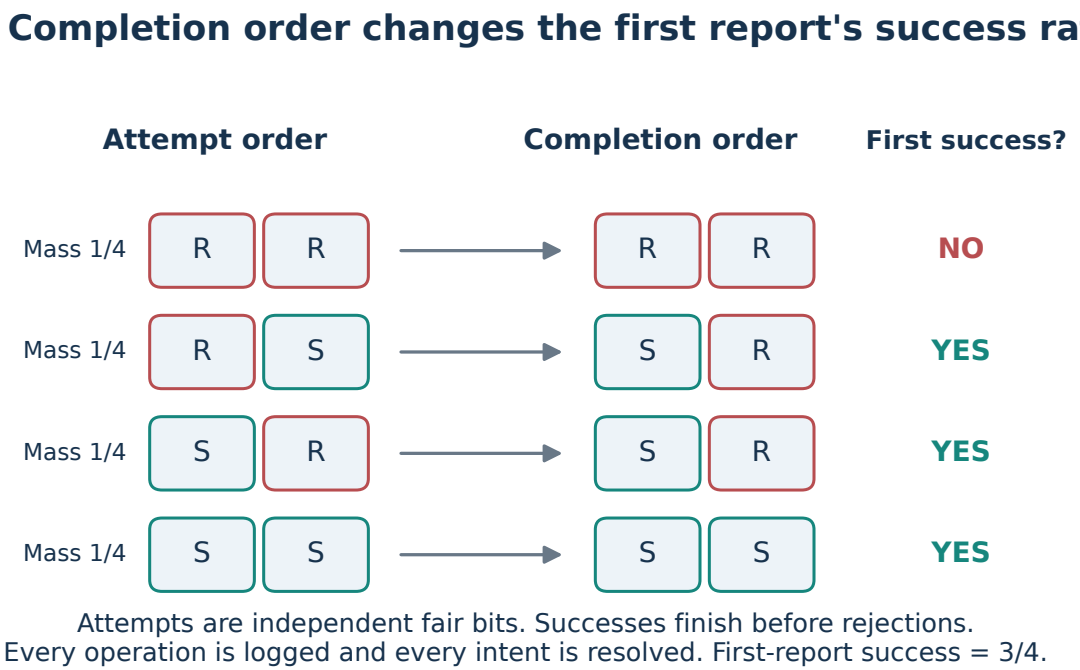


Figure 16: Of four equally likely initiated outcome pairs, three produce success as the first completed report under success-first ordering. Thus the first stored bit succeeds with probability $3/4$, although each attempted bit succeeds with probability $1/2$. Source: `results/v0.41.0/review_completion_order*`.

No observation has been deleted. No false label or duplicate is needed. Waiting until both attempts finish does not undo the biased first-completed prefix. The operation-ID lifecycle theorem can hold perfectly while the proposed statistical interpretation fails.

This obstruction localizes the missing bridge. The formal generative source is indexed in accepted report order. A physical attempt model indexed at initiation is a different object when latency depends on outcome. Independent attempted outcomes therefore do not by themselves justify IID first-completed samples.

An initiation-indexed protocol could be a route toward a valid physical model, but would need proved handling of pending holes, failures, replay and readiness. This guide does not claim that route is already completed.

Source: `FORMAL_THEORY.md`; `PROOF_BOUNDARY.md`; `CONTINUATION_v0.41.0.md`; `GAP_LEDGER.md`: G46.

27 The twenty-attempt example violates nominal calibration

Initiate twenty independent fair attempts, complete every success before every rejection, and resolve every intent. Examine the first ten completed reports. Whenever at least ten attempts succeed, that prefix is ten successes.

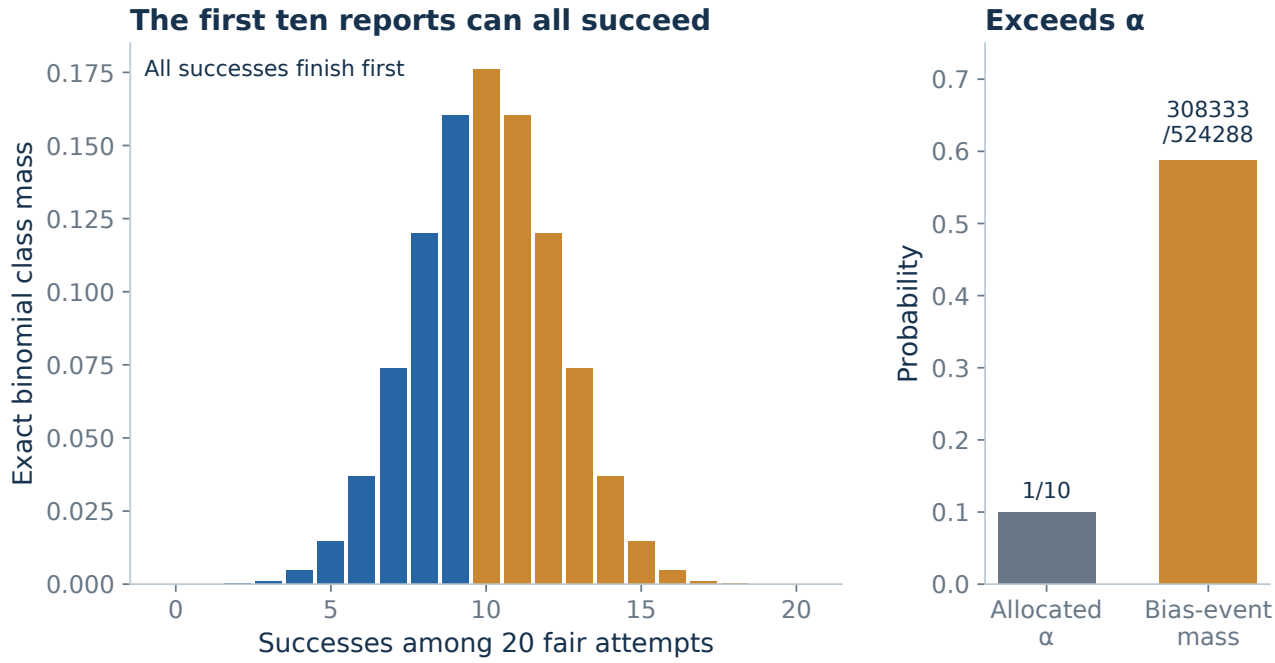


Figure 17: The event of at least ten successes has exact mass $308333/524288 \approx 0.588099$. On it, the implemented ten-success interval is $[3/5, 1]$, excluding the attempted parameter $1/2$. This event alone is above the nominal allowance $1/10$. Source: `results/v0.41.0/review_completion_order*`.

The exact lower bound is

$$\sum_{s=10}^{20} \binom{20}{s} 2^{-20} = \frac{308333}{524288} > \frac{1}{10}.$$

The independently enumerated total miss mass is $616667/1048576$, slightly larger because the zero-success case also returns an interval excluding $1/2$.

This is an exact finite computational counterexample to a proposed physical inference: complete, resolved initiation logs do not imply calibration of first-completed prefixes for the initiation probability. It is not a new Lean theorem and does not refute the defined source-generative theorem, whose report-order source law has not been satisfied here.

The lesson is practical: identity and lifecycle integrity are necessary components of trustworthy evidence, but statistical target and sampling order must also be specified.

Source: `FORMAL_THEORY.md`; `PROOF_BOUNDARY.md`; `results/v0.41.0/review_completion_order*`; `GAP_LEDGER.md`: G46.

28 Bayesian belief and calibration answer different questions

Bayesian inference updates a belief over hidden worlds using an observation likelihood. Finite-sample calibration bounds how frequently a confidence procedure excludes a fixed source parameter. Both appear in AVOC, with different mathematical roles.

For a finite rational prior b and channel E, an observation o with positive evidence mass gives

$$b_o(\omega) = \frac{b(\omega)E(\omega, o)}{\sum_{\omega'} b(\omega')E(\omega', o)}.$$

The archive connects posterior expected utility times predictive probability to the stochastic decision-value formula. This allows a decision calculation to be written either through posterior beliefs or directly through the observation channel.

A rejection can carry information. In the artifact workflow example, the prior probability of receipt availability is $1/2$. A coarse success report changes it to $3/4$; coarse rejection changes it to $1/4$. A particular finer rejection reason changes it instead to $1/3$. Ignoring rejection or conditioning on one finer reason while claiming to use the coarse channel would be incorrect.

Object	Its role
Bayesian posterior	Belief over worlds after the specified observation
Calibration band	Repeated-sampling coverage for a fixed source parameter
Artifact certificate	Exact semantic admission under a sound checker
Policy value	Expected utility of legal finite continuations

The hidden world in this workflow is receipt availability, not the truth of the proposition. Probabilistic confidence never replaces artifact admission.

Source: `AVOC/Bayesian.lean`: `bayesUpdate`, `bayesianPolicyValue_eq_stochasticDecisionValue`; `AVOC/ArtifactWorkflowExamples.lean`: `coarse_success_posterior`, `coarse_rejection_posterior`, `raw_rejection_posterior`.

29 The same first probability can hide different future behavior

The retry example has fee one and benefit four. Initially the success probability is $1/2$. One model has a persistent hidden reliability: fast worlds succeed with probability $9/10$ and slow worlds with probability $1/10$, with equal prior weights.

After a fault, the posterior fast weight becomes $1/10$ and slow weight $9/10$. The next mean success probability is

$$\frac{1}{10}\frac{9}{10} + \frac{9}{10}\frac{1}{10} = \frac{9}{50}.$$

Its expected immediate gain is $4 \cdot 9/50 - 1 = -7/25$, so the stated no-revival model abandons. Its value for every positive horizon from the initial state is one.

A second model has a fresh success probability $1/2$ at every attempt. A fault leaves the prior unchanged. With three attempts and the same credit, its value is $7/4$. The initial success marginal is identical, but the continuation value is different.

This shows why a state representation must preserve information relevant to future transitions, not merely today's mean success probability. Bayesian updates, stochastic channels and legal budget transitions meet in the continuation calculation.

The archive also gives an improving-service counterexample: a negative immediate action can become worthwhile over two steps if a fault improves future reliability. A myopic threshold therefore needs the no-revival hypotheses. Those hypotheses cannot be silently imported into every retry process.

Source: `AVOC/RetryExamples.lean`: `posterior_fast_after_fault`, `fault_resolve_probability`, `persistent_all_positive_horizons`, `iid_three_attempt_value`, `improving_two_steps_worthwhile`; `AVOC/RetryBayesian.lean`.

30 Rare evidence can amplify a tiny likelihood difference

Take an equal prior on two worlds, false and true. For any positive rational ε, compare likelihood weights $(\varepsilon, 0)$ with $(0, \varepsilon)$. They differ pointwise by at most ε, and both evidence masses are $\varepsilon/2$.

	Rare-false signal	Rare-true signal
Likelihood at false	ε	0
Likelihood at true	0	ε
Evidence mass	$\varepsilon/2$	$\varepsilon/2$
Posterior at false	1	0
Posterior at true	0	1

The posterior L^1 distance is two, its maximum possible value, for arbitrarily small ε. A small absolute likelihood error therefore cannot give a uniform small posterior error without a positive evidence-mass lower bound.

This is a different conditioning hazard from selecting only completed runs. Here the issue is numerical or representational instability of a rare Bayesian observation. In the completion example, the issue is changing the probability experiment by conditioning on a selected event. Both involve denominators that must be tracked explicitly.

A useful approximation certificate must retain the assumptions controlling its denominator. A statement that a channel is close pointwise is not enough by itself to guarantee every conditioned posterior is close. The archive proves the rare-evidence obstruction rather than hiding it behind a generic stability claim.

Source: `AVOC/BayesianInstability.lean`: `rare_evidence_maximal_posterior_flip`, `no_uniform_likelihood_lipschitz`; `AVOC/ApproximateBayesian.lean`.

31 What the finished chain establishes, and what comes next

The cumulative archive establishes a substantial defined chain: semantic representations and cost witnesses, exact artifact admission, legal finite-horizon planning, quote uncertainty propagation, selected-stage confirmation and recoverable records, finite-source calibration, and a lifecycle machine whose actual terminal samples feed a gated policy.

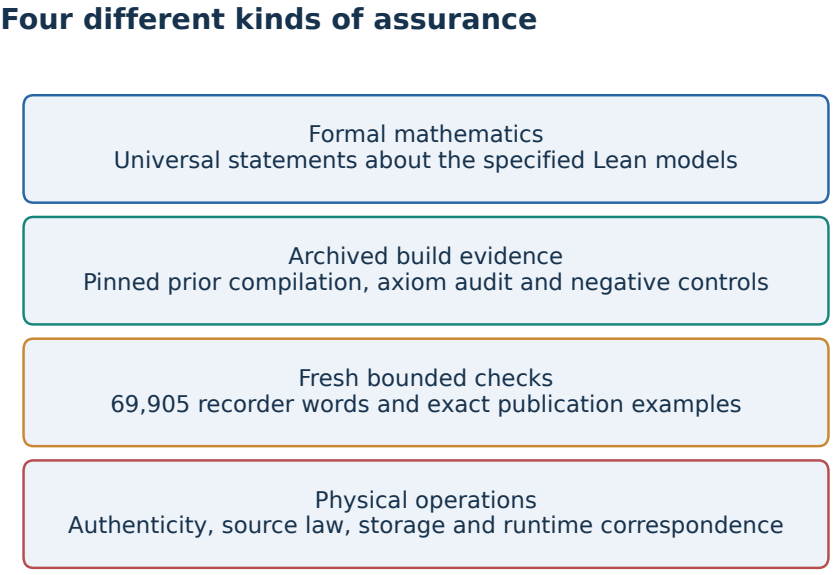


Figure 18: Three assurance layers remain distinct: proved formal claims, bounded executable correspondence, and assumptions about physical sources and external systems. Passing one layer does not automatically settle the others. Source: `PROOF_BOUNDARY.md`, `GAP_LEDGER.md` and `RECORDER_EXECUTABLE.md`.

The v0.41 final statistical claim is unconditional bad-issuance control under its finite rational conditional source law, fixed schedule, valid target strata and supplied deterministic memberships. Its inner episode certificate carries Q/value bounds and expected finite-horizon policy regret with frozen quotes and conservative action guards.

The pure recorder does not execute paid work, persist a journal, authenticate observations or prove physical exactly-once execution. Host/parser/storage correspondence is bounded evidence rather than a universal refinement proof. Real report-order coupling, empirical costs, drift, unrestricted stopping and cross-episode accumulation remain open.

A strong next experiment should choose one explicit physical sampling protocol and test its target, ordering and retention assumptions before claiming provider calibration. That would connect an established formal chain to a particular measured system, while retaining the exact counterexamples that show which proposed shortcuts fail.

Source: `FORMAL_THEORY.md`; `PROOF_BOUNDARY.md`; `GAP_LEDGER.md`: G44–G46; `CONTINUE_HERE.md`.