OPERATION KOMOREBI

REALIZATION CALCULUS



RESEARCH LIBRARY

AVOC

Artifact value, guarded operations, and calibrated planning

Traditional paper

Komorebi P.

Standardized publication edition / 8 October 2026

REALIZATION CALCULUS

Reading scope

A source-grounded exposition of the cumulative AVOC Formal v0.41.0 theory. The companion documents provide a worked visual route and a guide to the actual Lean declarations, evidence, and reproduction commands.

Abstract

An artifact can be correct and still be a poor computational investment. A probability estimate can be useful and still be invalid for the stream on which it is applied. A recovery protocol can preserve a logical checkpoint and still depend on an unverified external execution contract. AVOC develops these distinctions within a finite formal architecture for artifact planning. Its cumulative theory combines task sufficiency, stochastic and Bayesian information transport, exact checking for declared artifact languages, finite-horizon planning, robust quote bounds, rational finite-sample calibration, and recoverable state transitions. The latest recorder block connects complete executions of a source-generative lifecycle machine to its actual terminal sample prefixes and proves an unconditional bound on uncertified policy issuance. We present central definitions and proof mechanisms, worked exact examples, and a theorem-to-source map. We also explain a completion-order obstruction: honest and complete logging can turn independent attempted outcomes into biased reported prefixes. The publication is based on the preserved v0.41.0 sources and their archived Lean verification. Fresh source-inventory and exact Python checks are reported separately from compilation, which was not repeated for this edition.

Reading contract

This paper gives a mathematical exposition of the cumulative formal project, with explicit contracts for applying its results to external computational services. AVOC is the project name used throughout.

There are three complementary documents. The main paper develops the definitions, central arguments, and precise scope. The visual guide works through concrete examples with exact Python-generated figures. The Lean companion identifies actual definitions and theorem declarations, explains their hypotheses, and gives the pinned reproduction workflow. All source files, review notes, figure programs, and the original formal archive accompany the publication set.

The following evidence vocabulary is used throughout. A *formal result* refers to a statement in the delivered Lean theory with archived positive-build and axiom evidence. A *prose reconstruction* explains that argument in conventional notation. An *exact example* is a rational or finite computation with an explicit tested family. An *external premise* is an assumption about a provider, host, storage layer, cost conversion, or sampling process that is not derived from accepting a formal certificate. These categories are deliberately kept distinct.

Contents

1 Introduction: correct artifacts and correct decisions **4**

1.1 The central chain 4

1.2 What is established 5

1.3 Relation to classical mathematics 6

1.4 A running computational example 6

2 Representations, tasks, and retained distinctions 7
2.1 Query families and joint targets . . . 8

3 Information value and finite Bayesian evidence 9
3.1 Stochastic observations and garbling . . . 9
3.2 Exact Bayesian updates . . . 10
3.3 Compression must preserve the likelihood . . . 11

4 Sufficiency for controlled futures 11
4.1 Projected stochastic laws . . . 12

5 Artifacts, identities, and proof-carrying admission 13

6 Certified Boolean synthesis and semantic reuse 14
6.1 Programs and their declared work . . . 14
6.2 Bounded grammar search . . . 14
6.3 Compaction preserves the search optimum . . . 15
6.4 From bounded search to a global cost certificate . . . 16

7 Exact affine artifacts and complete finite probes 16
7.1 Typed normalization and safe contextual replacement . . . 17

8 Fusion, rank certificates, and retargeted reuse 17
8.1 Fusion has an amortization threshold . . . 18
8.2 An upper factorization and a lower rank witness . . . 18
8.3 Reuse under a changed offset . . . 19

9 Finite-horizon planning with admissible actions 19
9.1 Warm inspection and cold generation . . . 20
9.2 An executable evaluator and its cost boundary . . . 21

10 Quote uncertainty and certified lower-Q selection 21
10.1 From primitive bands to value bounds . . . 22
10.2 What the robust selector optimizes . . . 23
10.3 Whole-policy regret, not only first-action quality . . . 24

11 Exact finite-grid calibration 24
11.1 Predetermined schedules and data-dependent selection . . . 26

12 From source laws to actual terminal samples **26**
12.1 Pending intents and immutable terminal events 27
12.2 A source-generative recorder and whole-run refinement 27

13 Unconditional calibrated policy issuance **28**
13.1 Completion conditioning can select a biased population 29
13.2 Complete logs do not fix outcome-dependent completion order 29

14 Checked recovery and durable accounting **30**
14.1 Prepaid evaluation and upgrade journals 31
14.2 What recovery establishes about external execution 32

15 Evidence, reproduction, and the publication boundary **32**
15.1 Archived formal evidence 32
15.2 Fresh checks for this edition 33
15.3 Assurance matrix 33
15.4 The remaining research questions 34

16 Conclusion **35**

A Source and document navigation **35**

1 Introduction: correct artifacts and correct decisions

A useful computational system must decide more than whether an answer is correct. It must decide whether to search for an answer, inspect an existing candidate, reuse a checked result, or stop. It must account for checking and acquisition costs. When those costs and outcomes are uncertain, it must distinguish a plausible ranking from a certified decision. When work is interrupted, it must preserve enough state to resume without silently charging or installing the same logical result twice.

AVOC organizes these questions around three related objects: the semantics of an artifact, the value of an action, and the evidence supporting the action's quoted parameters. None can substitute for the others. An exact affine checker can establish that a stored factorization computes the desired map for all inputs. It cannot establish how likely a future provider is to return such a factorization. A confidence procedure can bound a failure event under a stated source law. It cannot establish that externally supplied log entries satisfy that law. A checkpoint theorem can establish idempotent acceptance of a logical commit. It cannot establish exactly-once execution of a physical operation.

This separation makes the architecture compositional. The artifact layer supplies a declared semantic language and checker. The decision layer supplies a state space, finite admissible action sets, finite observation alternatives, rewards, and transitions. The state space itself need not be finite. The uncertainty layer supplies frozen quote bands and compatible realizations. The observation layer supplies finite source laws, scheduled calibration, and a recorder whose terminal samples are connected to those sources. The recovery layer specifies how typed evidence and logical accounting are retained across replay.

1.1 The central chain

The central statistical composition has the following form. A finite rational probability space supplies per-identity latent Boolean sources satisfying the stated conditional fresh-outcome law. A predetermined finite schedule fixes the checkpoints and their error allowances. The source-generative recorder constructs terminal reports from those sources. A theorem identifies each observed sample list with the appropriate source prefix. Calibration then bounds failure of simultaneous scheduled probability-band validity. On the simultaneous-coverage event, every issued plan has the stated episode certificate. Outside that event a plan may be uncertified; the final theorem bounds the unconditional risk of that issuance.

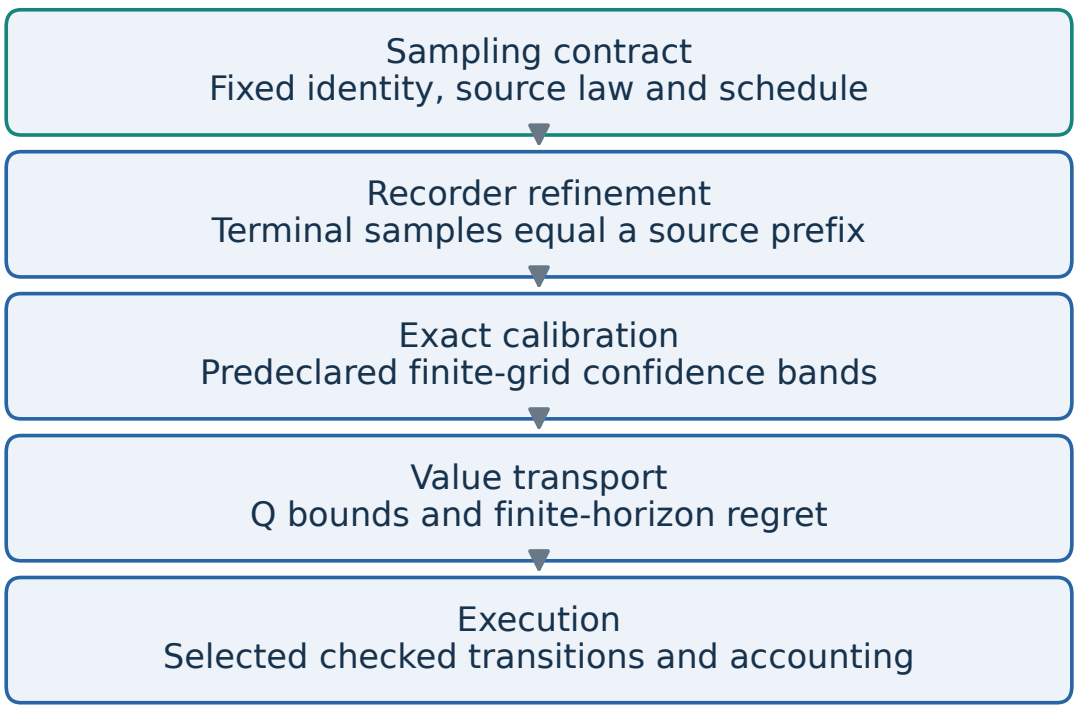


Figure 1: The dependency chain inside the declared model. Physical source integrity and durable host storage require their own justification. The chain is a source-based schematic, not a measurement.

Writing I for issuance and C for the specified episode certificate, the resulting conclusion is

$$\mathbb{P}(I \cap C^{\mathsf{c}}) \leq \sum_{j \in J} \alpha_j.$$

This is an unconditional probability over the original outcome space. It allows arbitrary dependence between scheduled checkpoints. It retains failed and unfinished runs in that space, with no plan issued on those runs. It does not assert a nominal bound conditional on issuance or completion, eventual issuance, or a productivity lower bound. The final recorder theorem derives the execution-to-prefix correspondence inside its generative model. The model's primitive source law remains an explicit premise.

1.2 What is established

The paper selects a coherent path through a larger cumulative library. The first group of results explains when a context or stochastic observation preserves what a task or decision problem needs. The second supplies complete checking and cost comparisons for declared Boolean and affine artifact languages.

The third proves finite planning and robust value guarantees under specified guards and tariffs. The fourth connects finite calibration, source laws, recorder executions, and optional policy issuance. The fifth preserves logical evidence and accounting under replay and recovery.

The global Boolean synthesis result is global within its declared finite semantic universe and syntax tariff. The affine minimum-width result concerns the specified two-stage factorization class. The policy guarantee concerns expected finite-horizon value against a comparator with the same conservative reservation guards. These restrictions are part of the conclusions, rather than qualifications attached after the fact.

1.3 Relation to classical mathematics

The formal development uses familiar mathematical principles: factorization through fibers, comparison of statistical experiments, finite conditional probability, dynamic programming, test inversion, union bounds, and linear algebra over a field. For example, composing a decision rule with a stochastic garbling proves the direction that a more informative observation cannot reduce attainable decision value. This belongs to the classical comparison-of-experiments tradition of Blackwell; the finite proof is given here and is not represented as a new general informativeness theorem.

The publication's mathematical focus is the integration of explicit contracts: which representation preserves a task, which artifact checker is complete for its language, which cost model supports a claimed saving, which uncertainty bound supports a selected action, and which sampling contract makes the calibration applicable. We make no exhaustive historical novelty claim. The names Bellman and Bayesian refer to the established planning and conditioning constructions used in the source, not to newly discovered replacements for those theories.

1.4 A running computational example

Consider an affine map over $\mathbb{F}_5$ with four inputs and four outputs, all-ones linear coefficient matrix, and offset $(1, 2, 3, 4)$. The direct dense evaluation tariff is $4(2 \cdot 4 + 1) = 36$. Its rank-one factorization has a $4 \to 1$ stage followed by a $1 \to 4$ stage, with tariff $9 + 12 = 21$. Thus each declared call saves 15 arithmetic cost units. For 1,000 calls the gross saving is 15,000.

The delivered exact-quote example gives warm inspection setup 133 and cold generation setup 196, both with success probability 1. Their net values are 14,867 and 14,804. These numbers are exact within the declared model. They are not wall-clock measurements, energy savings, or an empirical result about an external generation service. Later examples introduce uncertainty and show how certification can change an action, or lead a conservative planner to stop.

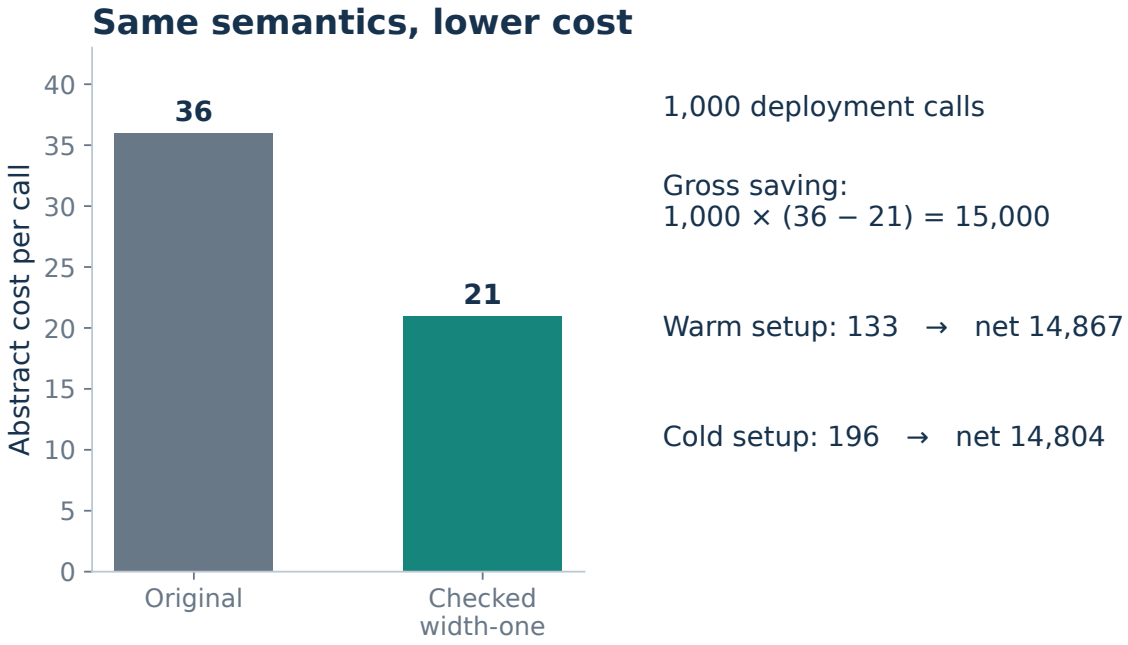


Figure 2: The rank-one affine example under its declared dense arithmetic tariff. Both the evaluation saving and the acquisition setup must enter the comparison.

The example gives one intelligible route through the whole architecture. The semantic certificate establishes the factorization. The tariff establishes a per-call saving. The quote identity binds that saving to a specified source, call count, checker, provider, and environment. The planner compares admissible acquisition actions. The recorder and calibration address the probability parameters only under their own source contracts. A recovery checkpoint retains a confirmed logical result without treating every replay as a fresh acquisition.

2 Representations, tasks, and retained distinctions

AVOC begins with an operational question about information: which distinctions must a retained artifact preserve so that a specified task can still be performed? The answer depends on the task. A representation that is adequate for one query can fail for another, and a representation that answers the current query can fail to support future controlled transitions. The formal development separates these obligations before assigning costs or probabilities.

Definition 2.1 (Deterministic refinement and task sufficiency)**.** Let X be a raw state space and let $\phi : X \to Z$ and $T : X \to Y$ be functions. We write

$$\phi \succeq T \quad \Longleftrightarrow \quad \exists d : Z \to Y \ \ \forall x \in X,\ d(\phi(x)) = T(x).$$

More generally, $\phi \succeq \psi$ means that ψ factors through ϕ. The relation is called refinement. The representation ϕ is task sufficient for T when $\phi \succeq T$. Two representations are information equivalent when each refines the other.

The decoder is part of the assertion. It may be computationally expensive, and its existence alone supplies no efficient implementation. Refinement is reflexive, using the identity decoder, and transitive, by com-

position of decoders. Thus it is a preorder. Information equivalence is the associated equivalence relation. Neither relation compares the byte length, acquisition cost, or evaluation time of the representations.

Theorem 2.2 (Exact fiber criterion)**.** *Suppose Y is nonempty. Then ϕ is sufficient for T if and only if*

$$\phi(x) = \phi(y) \quad \Longrightarrow \quad T(x) = T(y) \qquad (x, y \in X).$$

In particular, a single pair with $\phi(x) = \phi(y)$ and $T(x) \neq T(y)$ proves insufficiency.

Proof. If $T = d \circ \phi$, equality of the two representations gives equality after applying d. Conversely, fix a default $y_0 \in Y$. For every reachable $z \in \phi(X)$ choose x_z with $\phi(x_z) = z$ and set $d(z) = T(x_z)$. Set $d(z) = y_0$ on unreachable representation states. The fiber hypothesis ensures that $T(x_z) = T(x)$ whenever $z = \phi(x)$, so $d(\phi(x)) = T(x)$. The nonempty-codomain hypothesis supplies the default on unreachable states. The formal proof uses classical choice for the representatives. □

The target T itself is sufficient for T. Every sufficient representation refines T, by definition. Consequently T is a canonical coarsest sufficient representation in this information preorder. A sufficient representation is task minimal when T also refines it, and hence task minimality is exactly information equivalence with T. This is a statement about retained distinctions, rather than a theorem that the target can be computed cheaply or stored in a shortest file.

Proposition 2.3 (Finite data processing)**.** *For a finite sample $D \subseteq X$, let $c_D(\phi) = |\phi(D)|$ denote the number of reached representation states. If $\phi \succeq \psi$, then*

$$c_D(\psi) \leq c_D(\phi).$$

Proof. Choose d with $\psi = d \circ \phi$. Then $\psi(D) = d(\phi(D))$, and the image of a finite set under a function has cardinality at most that of the original set. □

The finite sample serves only to count reached states. Sufficiency in Theorem 2.2 is quantified over all of X. Agreement on a sample is therefore a different kind of evidence unless that sample exhausts the domain.

Example 2.4 (A task must be declared before compression)**.** Let $X = \{0,1\}^3$ and let $T(x_1, x_2, x_3) = x_1 \wedge x_2$. The representation $\phi(x) = T(x)$ is sufficient for this target, although it discards x_3 and many distinctions among the first two coordinates. The representation $\psi(x) = x_1 \oplus x_2$ is insufficient: $(0,0,0)$ and $(1,1,0)$ have the same ψ value and different T values. No decoder can repair this collision. Conversely, retaining all three coordinates refines T but retains more information than this task requires.

2.1 Query families and joint targets

Suppose a family of tasks is given by $T_q : X \to Y$, indexed by $q \in Q$. Define the joint target

$$T_{\text{joint}}(x) = (T_q(x))_{q \in Q} \in Y^Q.$$

Then ϕ is sufficient for the joint target if and only if it is sufficient for each individual task. The forward direction evaluates the decoded answer vector at q. For the reverse direction, choose a decoder d_q for each query and define $d(z)(q) = d_q(z)$. The formal theorem permits an arbitrary query index type, and the simultaneous decoder selection uses classical choice.

This distinction matters for reuse. An artifact can be minimal for one fixed target while a later target requires a distinction that the artifact discarded. A reusable library therefore needs a declared target family or an exact semantic key. Enlarging that family changes the sufficiency obligation. The framework does not infer the full future query family from the fact that past tasks succeeded.

3 Information value and finite Bayesian evidence

An information ordering becomes a decision ordering only after specifying a prior, available actions, and utility. The source treats the prior and utility as declared mathematical inputs. In probabilistic applications the prior is normalized and nonnegative. Some structural decision identities need less and are proved even for arbitrary rational weights.

Definition 3.1 (Deterministic experiment value)**.** Let Ω be finite, A be a finite nonempty action set, $b : \Omega \to \mathbb{Q}$ be a prior, and $u : \Omega \times A \to \mathbb{Q}$ be utility. For an observation $\phi : \Omega \to Z$ and policy $\pi : Z \to A$, define

$$J_b(\phi, \pi) = \sum_{\omega \in \Omega} b(\omega) u(\omega, \pi(\phi(\omega))).$$

When Z is finite, write $J_b^*(\phi) = \max_{\pi : Z \to A} J_b(\phi, \pi)$.

Theorem 3.2 (Deterministic information-value monotonicity)**.** *If $\phi \succeq \psi$, then $J_b^*(\psi) \leq J_b^*(\phi)$. If the acquisition costs satisfy $c_\phi \leq c_\psi$, then*

$$J_b^*(\psi) - c_\psi \leq J_b^*(\phi) - c_\phi.$$

Proof. Let d decode ψ from ϕ. Any coarse policy π lifts to $\pi \circ d$. For every world the lifted policy chooses exactly the same action, so its decision value is unchanged. A maximizing coarse policy exists because the policy space is finite and nonempty. Its lifted value is at most the best fine value. Subtracting the stated costs gives the second inequality. □

More information can increase gross decision value while decreasing net value if it costs more. The theorem's cost hypothesis is therefore essential to its domination conclusion. It also supplies a practical pruning rule: an experiment that is coarser and no cheaper can be removed from a finite decision menu without losing an optimum.

3.1 Stochastic observations and garbling

Definition 3.3 (Finite rational channel)**.** A channel from Ω to a finite observation alphabet O consists of rational entries $E(o \mid \omega) \geq 0$ with $\sum_o E(o \mid \omega) = 1$ for every ω. Given a second channel $K(y \mid o)$, their composite is

$$(EK)(y \mid \omega) = \sum_o E(o \mid \omega) K(y \mid o).$$

The composite is a garbling of E.

The unnormalized signal-action score and policy value are

$$S_E(o, a) = \sum_\omega b(\omega) E(o \mid \omega) u(\omega, a), \qquad J_b(E, \pi) = \sum_o S_E(o, \pi(o)).$$

This form avoids dividing by the evidence mass. In particular, it remains well defined on observations whose predictive probability is zero.

Theorem 3.4 (Finite stochastic information-value monotonicity)**.** *Every deterministic policy on EK is weakly dominated in value by some deterministic policy on E. Consequently the best deterministic value of EK is at most that of E.*

Proof. For each fine observation o, select $a_o \in A$ maximizing $S_E(o, a)$. Such a selection exists because A is finite and nonempty. For a coarse policy π, expand the channel composition and rearrange finite sums:

$$J_b(EK, \pi) = \sum_o \sum_y K(y \mid o) S_E(o, \pi(y)).$$

Every inner score is at most $S_E(o, a_o)$. Nonnegativity and normalization of the row of K give

$$\sum_y K(y \mid o) S_E(o, \pi(y)) \le S_E(o, a_o).$$

Summing over o proves domination by the fine policy $o \mapsto a_o$. Taking a best coarse policy yields the optimal-value comparison. □

Example 3.5 (Information has a price)**.** For two equally likely worlds, let the action predict the world and let utility be 1 for a correct prediction and 0 otherwise. A constant observation gives value $1/2$. Revealing the world gives value 1. If revelation costs $1/8$, its net value is $7/8$, exceeding the free constant experiment's $1/2$. If it costs $3/4$, its net value is $1/4$. The information order is unchanged across these two cases, while the preferred paid experiment changes.

3.2 Exact Bayesian updates

A rational belief is a nonnegative vector b on finite Ω with $\sum_\omega b(\omega) = 1$. For a channel E, define the predictive mass

$$p(o) = \sum_\omega b(\omega) E(o \mid \omega).$$

For $p(o) > 0$, the posterior is $b_o(\omega) = b(\omega)E(o \mid \omega)/p(o)$. The total updater in the source uses $b_o = b$ when $p(o) = 0$. This fallback is normalized and has zero coefficient in predictive expectations. Nonnegativity of every summand implies that each Bayes numerator also vanishes on such an impossible observation. Hence, with the total-update convention, the division-free identity

$$p(o) b_o(\omega) = b(\omega) E(o \mid \omega)$$

holds on every branch. It follows that

$$\sum_o p(o) \sum_\omega b_o(\omega) u(\omega, \pi(o)) = J_b(E, \pi).$$

Posterior-predictive evaluation and the unnormalized decision formula therefore agree exactly.

For any nonnegative likelihood weight L, write

$$M_b(L) = \sum_\omega b(\omega) L(\omega), \qquad C_L(b)(\omega) = \frac{b(\omega) L(\omega)}{M_b(L)}$$

when $M_b(L) > 0$. Given nonnegative L_1, L_2 and positive successive evidence masses, the source proves

$$M_b(L_1) M_{C_{L_1}(b)}(L_2) = M_b(L_1 L_2), \qquad C_{L_2}(C_{L_1}(b)) = C_{L_1 L_2}(b).$$

These identities follow by expanding the finite sums and canceling the positive normalizers. Both conditioning orders agree when their respective intermediate masses are positive. Applying this algebra to physical observations requires the appropriate joint likelihood model. In particular, multiplying likelihoods is justified by a stated factorization of the evidence law, rather than merely by the existence of two reports.

3.3 Compression must preserve the likelihood

Theorem 3.6 (Conditioning commutes with exact compression)**.** *Let $\phi : \Omega \to Z$ map between finite spaces, and define the pushforward belief $\bar{b}(z) = \sum_{\phi(\omega)=z} b(\omega)$. Suppose $L = \bar{L} \circ \phi$ for a nonnegative likelihood $\bar{L} : Z \to \mathbb{Q}$, and the evidence mass is positive. Then*

$$\phi_* C_L(b) = C_{\bar{L}}(\phi_* b).$$

Such an exact $\bar{L}$ exists if and only if L is constant on every fiber of ϕ.

Proof. Grouping the normalizer by fibers gives $M_b(L) = M_{\bar{b}}(\bar{L})$. On each fiber $\phi^{-1}(z)$, the likelihood can be factored outside the sum. Thus

$$\sum_{\phi(\omega)=z} \frac{b(\omega)L(\omega)}{M_b(L)} = \frac{\bar{b}(z)\bar{L}(z)}{M_{\bar{b}}(\bar{L})}.$$

The factorization criterion is Theorem 2.2 applied to the rational-valued target L. □

A prior compression that merges worlds with different likelihoods has no exact coarse likelihood for that observation. A later evidence update can consequently reveal a distinction that the compressed state no longer represents. This is the evidence version of the collision obstruction.

Example 3.7 (Rare evidence can reverse a posterior completely)**.** Under the uniform prior on $\{0,1\}$, take $L_0 = (\varepsilon, 0)$ and $L_1 = (0, \varepsilon)$ with rational $\varepsilon > 0$. The two likelihoods differ pointwise by at most ε, and both evidence masses equal $\varepsilon/2$. Their posteriors are respectively $(1,0)$ and $(0,1)$, with full L^1 distance 2. Arbitrarily small positive likelihood perturbations therefore do not imply uniform pointwise posterior stability without an additional evidence condition. The later model-error chapter uses weighted continuation comparisons rather than identifying nearby posterior points as the same controller state.

4 Sufficiency for controlled futures

For a one-shot task, refinement preserves sufficiency by decoder composition. A state abstraction for a controlled process has a second obligation: its fibers must remain compatible with the process's transitions. This extra obligation changes the mathematical behavior of refinement.

Definition 4.1 (Deterministic control sufficiency)**.** Let A_0 be a finite nonempty set of admissible actions, let $f : S \times A_0 \to S$ be transition, and let $r : S \times A_0 \to \mathbb{Q}$ be reward. A representation $\phi : S \to Z$ is control sufficient if, whenever $\phi(s) = \phi(t)$, every admissible action satisfies

$$r(s,a) = r(t,a), \qquad \phi(f(s,a)) = \phi(f(t,a)).$$

The global action set is common to all states in this definition.

Define the finite-horizon values by

$$V_0(s) = 0, \qquad Q_n(s,a) = r(s,a) + V_n(f(s,a)), \qquad V_{n+1}(s) = \max_{a \in A_0} Q_n(s,a).$$

Theorem 4.2 (Exact preservation of controlled value)**.** *If ϕ is control sufficient, then V_n is constant on every fiber of ϕ for every $n \in \mathbb{N}$. The same is true of $Q_n(\cdot, a)$ for each admissible a, and Bellman-optimal action sets coincide within each fiber.*

Proof. At horizon zero the values vanish. If V_n is fiber constant and $\phi(s) = \phi(t)$, control sufficiency gives equal rewards and equal represented successors. Hence $Q_n(s,a) = Q_n(t,a)$ for every admissible a. Taking the maximum over the common action set proves equality of V_{n+1}. The action-value equality also makes all inequalities defining optimal action membership identical at s and t. □

Example 4.3 (A finer representation can fail dynamic closure)**.** Consider $S = \{0,1,2\}$, one action, zero reward, and transitions $f(0) = 0$, $f(1) = 2$, $f(2) = 2$. The constant representation is control sufficient. The finer representation with fibers $\{0,1\}$ and $\{2\}$ refines that constant representation, but it merges states 0 and 1 whose successors fall into different fine fibers. It is therefore not control sufficient. This example is retained as an explicit formal counterexample to arbitrary refinement monotonicity.

The exact repair is simple: a refinement of a control-sufficient representation remains control sufficient if the finer representation is itself transition closed. Reward equality is inherited through the coarse decoder, while successor equality must be supplied for the fine representation. For models with state-dependent action sets, exact action feasibility is an additional obligation, handled by the guarded control interfaces later in the paper.

4.1 Projected stochastic laws

Let S be finite and replace deterministic transition by rational probability rows $P_a(s,s') \geq 0$ with $\sum_{s'} P_a(s,s') = 1$. The stochastic Bellman recursion is

$$V_0(s) = 0, \qquad Q_n(s,a) = r(s,a) + \sum_{s'} P_a(s,s')V_n(s'), \qquad V_{n+1}(s) = \max_{a \in A_0} Q_n(s,a).$$

Equality of entire raw transition rows is sufficient for compression, but can demand more than the abstract process needs. The appropriate condition compares their projected laws.

Definition 4.4 (Projected stochastic control sufficiency)**.** A representation $\phi : S \to Z$ is projected stochastically sufficient if equal represented current states have equal immediate rewards and, for every admissible a and every rational fiber-constant function $W : S \to \mathbb{Q}$,

$$\sum_{s'} P_a(s,s')W(s') = \sum_{s'} P_a(t,s')W(s') \quad \text{whenever } \phi(s) = \phi(t).$$

When Z is finite, this expectation condition is equivalent to equality of the mass sent to every abstract fiber:

$$\sum_{\phi(s')=z'} P_a(s,s') = \sum_{\phi(s')=z'} P_a(t,s') \qquad (z' \in Z).$$

To prove necessity, use the indicator of the fiber as W. To prove sufficiency, group the expectation by fibers and factor out the common value of W on each nonempty fiber. Empty fibers contribute zero. These arguments explain why raw states may have distinct next-state probabilities while having identical controlled abstract behavior.

Theorem 4.5 (Exact stochastic quotient)**.** *Suppose $\phi : S \to Z$ is surjective and projected stochastically sufficient, with S, Z finite. Choose a representative s_z of every fiber, and define*

$$\bar{r}(z,a) = r(s_z,a), \qquad \bar{P}_a(z,z') = \sum_{\phi(s')=z'} P_a(s_z,s').$$

These quantities give a normalized rational stochastic control system, independent of the representative choices for admissible actions. For every finite horizon,

$$\bar{V}_n(\phi(s)) = V_n(s).$$

Proof. Nonnegative fiber masses give nonnegative quotient probabilities; partitioning S by the fibers shows that each quotient row sums to one. Sufficiency makes reward and projected mass unchanged when replacing s_z by another representative. Inductively assume the value equality at horizon n. Then

$$\sum_{z'} \bar{P}_a(\phi(s), z')\bar{V}_n(z') = \sum_{s'} P_a(s, s')V_n(s'),$$

by projected equality and fiberwise grouping. Immediate rewards agree, so every action backup agrees. Taking the maximum over the common action set proves the next horizon. The zero-horizon case is immediate. □

This result concerns exact finite models with a specified reward and transition law. It neither discovers the correct abstraction automatically nor identifies a supplied rational model with a physical system. Its role is to certify the consequences of a declared abstraction once the one-step obligations have been established.

5 Artifacts, identities, and proof-carrying admission

An artifact's semantic content and its operational identity serve different purposes. Semantic equivalence permits replacement in a declared task language. Operational identity prevents a receipt for one statement, dependency set, environment, or generation from authorizing another.

The concrete binding layer uses a small data-only propositional language. Formulas contain numbered atoms, truth, falsity, conjunction, disjunction, and implication. Derivations contain the corresponding introduction and elimination rules. Atoms are interpreted as arbitrary propositions under a valuation $v : \mathbb{N} \to \text{Prop}$. A recursive inference function either returns a formula or rejects. It accepts data representing a derivation, rather than executable tactics or imported source code.

Theorem 5.1 (Checker soundness)**.** *If the inference checker accepts derivation p for target F under context Γ, then every valuation satisfying Γ satisfies F.*

Proof. Induct on the derivation. A used assumption belongs to Γ and is true by the context hypothesis. Conjunction introduction combines the two inductive conclusions, and its projections select the corresponding conclusion. Implication introduction applies the induction hypothesis in the extended context; implication elimination applies a proved implication to a proved premise. The two disjunction injections construct the appropriate alternative. Case elimination applies the matching branch proof after extending the context with the observed alternative. Falsity elimination follows from the impossible proposition. The checker's shape and equality tests ensure that the formulas in these rules match exactly. □

This is soundness of the defined propositional checker. It is not a universal ingestion theorem for arbitrary Lean source. Dependencies remain assumptions in the checker context, and their truth must come from the surrounding certified workflow.

Definition 5.2 (Protected artifact identity)**.** An identity records a task name, a formula, its statement-byte snapshot, a list of dependencies with their formulas and artifact-byte snapshots, and an environment snapshot. An artifact contains that identity, a source-byte snapshot, and a data-only derivation. A ticket consists of a generation number together with an artifact. A receipt names the exact ticket and records an approval bit.

All byte snapshots in this model are exact strings. No finite hash is assumed injective. The validator checks four obligations: the pending artifact's identity equals the current identity, the receipt names the exact pending ticket, its approval bit is true, and the derivation checks for the current target under the current dependency context. The resulting validation theorem supplies the target proposition whenever the dependency propositions hold.

A changed statement, dependency, environment, source snapshot, or ticket generation is rejected by the corresponding equality gate. Approval alone supplies no proof. Re-registering an identical payload still changes its generation, which prevents an old receipt from serving as authorization for a new registration.

The finite slot model increments its generation on registration. Successful admission consumes the pending ticket; failed admission leaves it unchanged. A successfully consumed receipt cannot admit again because no pending ticket remains. These statements describe admission in the defined reducer. They do not establish exactly-once performance of external paid work, filesystem durability, or authentication of an external provider. The recovery chapters treat the separate accounting and persistence obligations.

6 Certified Boolean synthesis and semantic reuse

The Boolean backend provides a complete finite example of semantic checking, cost-aware replacement, and retained computation. Its restricted language is deliberate: strong universal conclusions are possible because both the input domain and semantics are explicit.

6.1 Programs and their declared work

Programs are finite abstract syntax trees built from Boolean literals, three input variables, negation, conjunction, disjunction, and exclusive-or. The input domain is the eight triples in $\{0,1\}^3$. Let $N(p)$ be the tree-node count: each literal or variable contributes one, negation adds one to its child, and a binary constructor adds one to the sum of its child counts.

The instrumented evaluator evaluates both children of a binary operation. A structural induction proves that its output equals the ordinary evaluator and that its visit count is exactly $N(p)$ for every input. Thus node count is a valid eager-traversal meter for this interpreter. It is not a short-circuit meter, hardware runtime, memory-allocation count, or Boolean circuit size with shared subexpressions.

Theorem 6.1 (Complete Boolean equivalence check)**.** *Let $\sigma(p)$ be the eight output bits of p, in the fixed lexicographic input order. Then*

$$\sigma(p) = \sigma(q) \quad \Longleftrightarrow \quad \forall x \in \{0,1\}^3,\ p(x) = q(x).$$

The executable checker comparing all eight outputs is sound and complete.

Proof. Equality of signatures gives equality at every entry. Every possible input occurs in the enumerated list, so every input is covered. Conversely, pointwise equality makes the two mapped output lists equal. This is exhaustive finite checking, not a sampled approximation. □

6.2 Bounded grammar search

The initial grammar layer has five atoms: two literals and three variables. If a layer is L, expansion retains L, adds all negations, and adds all ordered pairs under each of the three binary operations. Let

M_d count emitted list entries at depth d. The source proves

$$M_0 = 5, \qquad M_{d+1} = 2M_d + 3M_d^2.$$

The entries can contain duplicates. The node bound is $B_0 = 1$, $B_{d+1} = 2B_d + 1$, and every expression of height at most d occurs in the depth-d list. In particular $M_1 = 85$ and $M_2 = 21845$.

The search retains an incumbent equivalent to a reference. A candidate replaces it only when the complete checker passes and the candidate has strictly fewer nodes. Induction over the examined list proves preservation of equivalence and nonincrease of node count. The returned incumbent has node count at most that of every equivalent examined candidate.

Generation is reserved before a layer is constructed. The declared generation charge is the sum of the layer-entry counts through the chosen depth. A candidate comparison is conservatively charged

$$T(r, d) = 8\big(N(r) + B_d + 1\big).$$

The examined prefix is limited by both a user trial cap and the remaining budget divided by $T(r, d)$. Its generation plus examination charge is proved at most the supplied budget. If the generation charge does not fit, search returns the original reference and incurs zero modeled search charge. If the entire layer fits both limits, the result is optimal among equivalent expressions of height at most d. A partial prefix provides optimality only over that prefix.

Example 6.2 (A smaller artifact need not repay its discovery)**.** The reference $\neg(\neg x_1 \vee \neg x_2)$ has six nodes. The equivalent conjunction $x_1 \wedge x_2$ has three. The archived depth-one search examines the complete 85-entry layer under its stated limits and returns the conjunction, with modeled search charge 6890. Each eager evaluation subsequently saves three visits. A large deployment can repay the exploration cost; a single evaluation cannot. The synthesis theorem supplies semantic safety and a budget bound even in the unprofitable case, and the promotion layer separately checks the campaign economics.

6.3 Compaction preserves the search optimum

Semantic entries carry an expression, its complete eight-bit signature, its node count, and equations certifying the two cached fields. The index retains at most one representative of each signature and keeps a smallest representative among inserted expressions. There are at most $2^8 = 256$ entries, regardless of the size of the submitted list.

Proposition 6.3 (Recursive semantic coverage)**.** *For every depth d, recursively expanding the retained representatives and compacting again preserves the smallest equivalent node count obtainable from the full uncompressed grammar layer at that depth, with the same reference fallback.*

Proof. Compaction gives every input expression an equivalent retained representative of no greater node count. Equivalence is a congruence for each grammar constructor. Replacing either child by a no-larger equivalent representative therefore preserves the result and does not increase its node count. Negation has the same property. Induction on depth supplies semantic and cost coverage of the full grammar by recursive representatives. Conversely, every retained representative originates in the full grammar. These two inclusions in semantic-cost coverage give equal selected optimum costs. □

This is a precise example of reusable knowledge reducing subsequent construction. It is finite semantic compaction with a proof of optimum preservation. It does not establish that arbitrary learned representations preserve future optimization problems. The instrumented construction separately records emitted

entries, total node mass, and index probes; its declared tariff excludes several host costs, including allocation and serialization.

6.4 From bounded search to a global cost certificate

A complete finite semantic algebra supports a stronger conclusion than searching a deeper finite layer. Let D be a semantic domain with interpretations of the Boolean constructors. A cost function $c : D \to \mathbb{N}$ is a lower certificate if it satisfies the constructor inequalities

$$c(\text{literal } b) \le 1, \qquad c(\text{variable } j) \le 1,$$
$$c(\neg s) \le c(s) + 1, \qquad c(s \circ t) \le c(s) + c(t) + 1 \quad (\circ \in \{\wedge, \vee, \oplus\}).$$

Theorem 6.4 (Attained global semantic minimum)**.** *Suppose c satisfies those inequalities and, for every semantic state s, a witness expression w_s has semantics s and node count $c(s)$. Then for every expression p with semantics s,*

$$N(w_s) = c(s) \le N(p).$$

No expression-height bound appears in this conclusion. Two such attained cost certificates have identical cost functions.

Proof. Structural induction proves $c(\mathrm{sem}(p)) \le N(p)$. The leaf inequalities give the base cases. For a unary constructor, use its inequality and the child's induction hypothesis. For a binary constructor, use its inequality and both child hypotheses. Applying this lower bound to any expression with semantics s and using the attained witness proves global minimality. For two certificates, apply each one's lower bound to the other's witness, obtaining both inequalities between their costs. □

For the three-input Boolean backend, D consists of all 256 truth signatures. A proposed bank must have exactly 256 rows. Each row's stored expression is checked for its exact signature and claimed node count, and the bank is checked against every unary and binary constructor inequality. An accepted bank therefore carries the hypotheses of Theorem 6.4. Its query returns a globally node-minimal expression within this entire Boolean tree grammar, including all finite heights.

The distinction is substantive. A bounded search is global only within its examined language fragment. A checked closure bank certifies a lower bound for every expression by structural induction and supplies witnesses attaining that bound. The universal conclusion remains specific to the grammar, three-input semantics, and eager tree-node tariff. It does not claim unrestricted program minimization or optimal implementation on physical hardware.

7 Exact affine artifacts and complete finite probes

The affine backend extends exact semantic certification to infinite input domains. Its complete finite test is possible because the represented functions have a fixed algebraic form.

Definition 7.1 (Dimension-indexed affine signature)**.** Over a ring R, a signature from dimension n to dimension m is a pair (A, b) with $A \in R^{m \times n}$ and $b \in R^m$, denoting $x \mapsto Ax + b$. Dimensions are part of the type. Composition and pointwise addition are

$$(B, c) \circ (A, b) = (BA, Bb + c), \qquad (A, b) + (A', b') = (A + A', b + b').$$

Theorem 7.2 (Complete zero-and-basis probes)**.** *Two affine signatures $f, g : R^n \to R^m$ are equal if and only if they agree at 0 and at every standard basis vector e_j. Thus $n+1$ vector probes establish equality on all inputs. A mismatch in signatures has a mismatch among these probes.*

Proof. The zero probe identifies the offsets because $f(0) = b_f$. At e_j, the output is the jth matrix column plus the offset. After canceling the now-equal offsets, equality of the basis probes identifies every column. Equal coefficients give equal evaluations for all inputs. The mismatch assertion is the contrapositive. □

The theorem is algebraic completeness, rather than empirical interpolation from arbitrary function values. Its premise is that both candidates belong to the affine language over the stated ring. Adding a nonlinear primitive changes the language and invalidates this probe argument unless a new completeness theorem is supplied.

Example 7.3 (The zero probe is necessary)**.** Over $\mathbb{Q}$, consider scalar maps $f(x) = x$ and $g(x) = 2x - 1$. They agree at the single basis vector $x = 1$, but $f(0) = 0$ and $g(0) = -1$. Omitting the zero probe would accept a false affine equivalence. Over a field, equality of the linear matrices alone also leaves arbitrary affine offsets unconstrained.

7.1 Typed normalization and safe contextual replacement

Typed programs combine affine atoms, identities, sequential composition, and pointwise addition. Composition connects only matching intermediate dimensions. An independent recursive execution semantics and coefficient-normalization semantics are defined. Structural induction proves

$$\mathrm{eval}(\mathrm{normalize}(p), x) = \mathrm{run}(p, x) \qquad \text{for all } x.$$

Hence two programs are equivalent if and only if their normal signatures coincide. Decidable coefficient comparison is a sound and complete checker when equality in R is decidable, as it is for the rational and finite-field executable examples.

One-hole typed contexts permit replacement at arbitrary nesting depth. Equivalence is preserved under composition and pointwise addition, so plugging equivalent programs into such a context preserves its semantics. A declared additive nonnegative primitive tariff, with m charged for adding two m-dimensional outputs, also makes contextual cost monotone: a no-more-expensive replacement cannot increase the whole context's declared cost. This is a compositional guarantee for the explicit syntax and tariff.

8 Fusion, rank certificates, and retargeted reuse

An exact normal form can improve verification without improving evaluation. The affine backend therefore checks semantics and cost separately. Its dense arithmetic tariff is

$$D(n, m) = m(2n + 1),$$

charging n multiplications, n accumulations, and one bias addition per output row. The coefficient-composition tariff is

$$C(n, m, k) = 2knm + D(m, k).$$

These are declared scalar-operation tariffs. They exclude bit complexity, memory traffic, allocation, and physical timing.

8.1 Fusion has an amortization threshold

Replacing a composed program by one dense atom with its normalized coefficients preserves every output. Its per-call execution tariff becomes $D(n, m)$. The source's no-slower selector retains the original program whenever that dense tariff is greater than its current execution cost. Normalization work is charged separately for campaign comparisons.

Example 8.1 (Beneficial and harmful fusion)**.** A dense $2 \to 3 \to 2$ chain has per-call tariff $D(2,3) + D(3,2) = 29$. Its fused $2 \to 2$ map costs 10 per call, and coefficient composition costs 38. Thus $38 + 10q \leq 29q$ exactly when $q \geq 2$, with strict savings exactly when $q \geq 3$.

In contrast, a $4 \to 1 \to 4$ bottleneck costs $D(4,1) + D(1,4) = 21$ per call, while its dense fused $4 \to 4$ form costs 36. Fusion remains semantically correct but loses its cheap intermediate bottleneck. The cost-aware selector therefore keeps the original chain in that comparison.

The second case motivates retaining a factorized artifact. A dense semantic key can identify a function while a different execution representation realizes it more cheaply.

8.2 An upper factorization and a lower rank witness

Work over a field R. For reference $f = (A, b) : R^n \to R^m$, a candidate supplies an intermediate width k, first and second affine signatures

$$a(x) = Ux + u_0, \qquad d(y) = Vy + v_0,$$

and matrices $L \in R^{k \times m}$, $H \in R^{n \times k}$. The checker verifies

$$VU = A, \qquad Vu_0 + v_0 = b, \qquad LAH = I_k.$$

The first two identities give the exact upper implementation. The third is an independently checkable lower witness. Validation uses polynomial matrix identities; an untrusted rank routine is not part of the logical justification.

Theorem 8.2 (Certified minimum affine width)**.** *An accepted candidate has $k = \operatorname{rank}(A)$. Every exact two-stage affine factorization of f through width ℓ satisfies $k \leq \ell$. Under the dense two-stage tariff,*

$$F(n, m, k) = D(n, k) + D(k, m),$$

the accepted candidate is no more expensive than any such two-stage factorization.

Proof. The upper factorization gives $\operatorname{rank}(A) = \operatorname{rank}(VU) \leq k$. The lower witness gives

$$k = \operatorname{rank}(I_k) = \operatorname{rank}(LAH) \leq \operatorname{rank}(A).$$

Thus rank equals k. Any exact factorization through ℓ has $\operatorname{rank}(A) \leq \ell$, by the same upper rank inequality. Finally,

$$F(n, m, k) = k(2n + 1) + m(2k + 1)$$

is increasing in k, so minimum width gives minimum tariff in this class. □

Exact factorization alone does not prove minimum width. A redundant factorization can be correct while failing to supply a valid $LAH = I_k$. Conversely, a lower rank witness alone does not prove that the proposed affine implementation has the correct offset. Both directions are needed.

Example 8.3 (A checked rank-one bottleneck)**.** In the archived finite-field example over $\mathbf{F}_5$, A is the 4×4 all-ones matrix and $b = (1, 2, 3, 4)$. The first stage computes $y = x_1 + x_2 + x_3 + x_4$. The second emits $(y + 1, y + 2, y + 3, y + 4)$. Taking L to select the first output coordinate and H to select the first input coordinate gives $LAH = (1)$. The width-one factorization is therefore exact and width minimal.

Its tariff is $9 + 12 = 21$, compared with dense cost 36, saving 15 per call. The declared factor-checking charge is 105. For this already atomic reference, the campaign inequality is exactly

$$105 + g \leq 15q,$$

where g is the declared generation work and q the deployment count. With $g = 0$, seven calls break even. Generation, validation, and deployment are all visible in this comparison.

The menu selector compares three equivalent implementations: the current program, its dense fused atom, and the accepted minimum-width two-stage plan. Its selected tariff is exactly

$$\min\{\mathrm{cost}(p),\ D(n, m),\ F(n, m, k)\}.$$

This proves a minimum over the explicit menu, and also comparison with every exact two-dense-stage factorization. It does not prove a globally cheapest affine program in the larger language. Sparse schedules, sharing, special primitives, and hardware-specific algorithms have different admissible implementations or tariffs. An identity program already costs zero in this syntax and remains preferable to a paid dense factorization.

8.3 Reuse under a changed offset

Suppose a stored candidate has been validated for (A, b) and a later target is (A, b') with exactly the same linear matrix. Keep U, u_0, V, L, H and replace only the second-stage offset by

$$v_0' = b' - Vu_0.$$

Then $VU = A$ is unchanged, $Vu_0 + v_0' = b'$, and $LAH = I_k$ remains true. The same width and two-stage tariff are retained. A second retargeting overwrites the offset rather than accumulating prior offset changes.

The import gate rechecks the entire stored certificate and exact equality of the current linear semantic key before constructing the retargeted checked record. It rejects invalid saved certificates and changed linear matrices. The current implementation is then compared with the reused plan, so a previously beneficial plan is not automatically preferred after other implementation changes.

This is an exact reuse theorem for a specified invariant key. It shows which part of the calculation is reusable and which part must be adapted. Persisting a claimed certificate, trusting a previous receipt, or comparing a suggestive task label would not provide these hypotheses. The proof-carrying bank and the protected artifact workflow supply complementary mechanisms for making that distinction explicit.

9 Finite-horizon planning with admissible actions

An artifact decision changes both what is available and what can be afforded. Inspection can turn an untried cached artifact into a ready artifact, but a rejection also spends its inspection fee. Generation can create a candidate, but deployment must wait for a checked success. A controller therefore needs an action set that depends on the current state, rather than a fixed menu with inadmissible actions assigned convenient scores.

Definition 9.1 (Guarded rational system)**.** Let S be a state space, $A(s)$ a finite nonempty set of actions, and O a finite observation space. A guarded system consists of a rational reward $r(s,a)$, a transition $T(s,a,o)$, and rational probabilities $P(o \mid s,a)$ satisfying

$$P(o \mid s,a) \geq 0, \qquad \sum_{o \in O} P(o \mid s,a) = 1.$$

Its finite-horizon Bellman values are

$$V_0(s) = 0, \tag{1}$$
$$Q_h(s,a) = r(s,a) + \sum_o P(o \mid s,a) V_h(T(s,a,o)), \tag{2}$$
$$V_{h+1}(s) = \max_{a \in A(s)} Q_h(s,a). \tag{3}$$

Here h in Q_h counts *continuation* steps. Choosing an acquisition action and then deploying a successful artifact requires two total actions, so its first-action score is Q_1, not Q_0. This indexing also applies to `robustChoice md h s` in the formal source.

The state space need not be finite. Rational Bayesian beliefs are one intended instance. Only each feasible action set and the observation space are finite. The finite maximum exists even when there is no asserted efficient procedure for finding it. The mathematical selector and the executable evaluator are separate constructions.

Theorem 9.2 (Optimality over complete adaptive plans)**.** *Every horizon H has a legal deterministic observation tree attaining $V_H(s)$. Every other legal deterministic observation tree has value at most $V_H(s)$. If each admissible transition preserves an invariant, the attaining tree preserves that invariant on every branch.*

Proof. At horizon zero, the empty tree has value zero. At horizon $h+1$, choose an action attaining the finite maximum in (3); after each observation, attach an optimal horizon-h tree at the successor state. Induction proves equality of the resulting expected value and the Bellman value. For an arbitrary legal tree, induction bounds each child value by the corresponding optimal continuation. Multiplication by nonnegative probabilities, summation, and the maximum over admissible first actions give the upper bound. Invariant preservation follows by the same tree induction. The invariant argument applies to every observation branch, including a branch of probability zero. □

This is a prose reconstruction of `safe_optimal_plan_exists`, `optimalPlan_value`, and `planValue_le_value` in `AVOC/GuardedControl.lean`. The competitors are complete adaptive observation trees. They need not use the same subsequent choices as the selected controller. The comparator nevertheless uses the same state, information, action guards, and transition model. An oracle with additional information is a different optimization problem.

9.1 Warm inspection and cold generation

The lazy staged controller makes this structure concrete. Its state records warm and cold statuses, remaining budget, reserved spending, original credit, and a terminal flag. Each artifact status is untried, failed, or ready. The action list always contains stop. Inspection and generation require an untried status and enough budget. Deployment requires the matching ready status. Stopping or deploying ends the episode.

An acquisition transition pays its setup fee whether its Boolean report is true or false. A true report changes the corresponding status to ready. A false report changes it to failed. Acquisition rewards are negative setup fees, deployment rewards are nonnegative gross values, and stopping has reward zero. This prevents three easy accounting mistakes: assuming unsuccessful acquisition is free, treating a quote as an already available artifact, and deploying before checked success.

Example 9.3 (The artifact-backed rank-one comparison)**.** The exact quote model retained in `AVOC/RobustQuoteExamples.lean` has warm setup 133, cold setup 196, success probability 1 for both, and gross deployment value 15000 for each over 1000 calls. With initial credit 196 and two total actions available,

$$Q_1(\text{inspectWarm}) = 15000 - 133 = 14867, \qquad Q_1(\text{generateCold}) = 15000 - 196 = 14804.$$

Warm inspection is better by 63 in this exact model. At one total action, acquisition has only its negative immediate reward, and stopping is preferable when neither artifact is ready. The deployment opportunity creates the change between these two horizons.

This example has actual bounded artifact witnesses in the retained source. The gross value is the symbolic execution saving for the protected workload, not a measured dollar return or a wall-clock benchmark. The singleton evidence makes all quote uncertainty radii vanish, so the robust lower-Q selector recovers the inherited exact selector. The relevant declarations are `exact_recovers_v037`, `exact_all_local_radii_zero`, and `artifact_refines_deterministic_fields`.

9.2 An executable evaluator and its cost boundary

The executable planner enumerates admissible first actions and observation branches, recursively evaluates continuations, and chooses a best score. Under the formal `Realizes` relation, its returned score equals the mathematical Bellman value. A full catalogue enumeration and a verified admissibility check connect the staged workflow to that evaluator. In particular, `WorkflowPlanner.model_realizes` and `executable_policy_source_safe` connect actual staged transitions, source-bound artifacts, and the selected policy.

The planner's work counter counts defined recursive evaluator nodes. A branching cap permits reserving an abstract planning allowance, and truncated lookahead has a source-backed policy regret bound. This counter is not a rational-bit complexity bound. It does not include all host parsing, catalogue scans, reservation arithmetic, storage operations, or physical invocation cost. Consequently, a proved allowance in that model cannot by itself establish a time bound for the complete service.

10 Quote uncertainty and certified lower-Q selection

Quotes carry more than a success estimate. A protected identity records the artifact kind, exact source signature, workload call count, cost model, provider, checker, environment, and schema. A setup band has natural endpoints. Gross and probability bands have rational endpoints, with probability endpoints in $[0, 1]$ and nonnegative gross lower endpoints. A submitted quote validates only when it equals the protected approved quote. Changing either identity or any band changes the evidence.

For kind $k \in \{w, c\}$, write

$$c_k \in [c_k^-, c_k^+], \qquad g_k \in [g_k^-, g_k^+], \qquad p_k \in [p_k^-, p_k^+].$$

The formal nominal model reserves c_k^+, credits g_k^-, and uses p_k^-. A compatible realization supplies exact values inside these bands. In the true comparison system, rewards and report probabilities use those

exact values, but the state transitions and action guards remain those of the nominal model. In particular, setup reservations continue to use the upper setup bound.

This common guard matters. If the true setup is 1 and the upper quoted setup is 3, a budget of 2 still makes the acquisition inadmissible for both systems in the theorem. The comparator does not obtain an extra feasible action by knowing the smaller true fee. A theorem against that larger feasible set would require a different comparison argument.

10.1 From primitive bands to value bounds

Let $w_c(k) = c_k^+ - c_k^-$, $w_g(k) = g_k^+ - g_k^-$, and $w_p(k) = p_k^+ - p_k^-$. The source uses the uniform quantities

$$\varepsilon = w_c(w) + w_c(c) + w_g(w) + w_g(c), \tag{4}$$

$$\delta = 2\big(w_p(w) + w_p(c)\big), \tag{5}$$

$$M = c_w^+ + c_c^+ + g_w^+ + g_c^+. \tag{6}$$

They bound reward error, Boolean-row L^1 error, and nominal one-step reward magnitude respectively. For an acquisition of kind k, the sharper local reward error is $w_c(k)$ and its row error is $2w_p(k)$. For deployment of kind k, the local reward error is $w_g(k)$ and its row error is zero. Stop has zero primitive errors.

Lemma 10.1 (Bernoulli row error and expectation transport)**.** *For Bernoulli parameters p, q,*

$$\sum_{o\in\{0,1\}} |P_p(o) - P_q(o)| = 2|p - q|.$$

If $|W(t)| \le B$, then replacing an observation row P by $\widehat{P}$ changes its expected continuation by at most

$$\left|\sum_o (P(o) - \widehat{P}(o))W(T(s,a,o))\right| \le B\sum_o |P(o) - \widehat{P}(o)|.$$

Proof. The two Bernoulli row differences are $p - q$ and $q - p$, which have equal absolute values. For the second statement, apply the triangle inequality to the finite sum and use the uniform bound on each continuation value. □

Induction on the Bellman recursion gives $|\widehat{V}_h(s)| \le hM$ for the nominal value. The source's accumulated value radius and local Q radius are

$$D_h = \sum_{j=0}^{h-1} (\varepsilon + \delta j M), \tag{7}$$

$$\rho_h(a) = D_h + \varepsilon_a + \delta_a h M, \tag{8}$$

$$\Gamma_h = D_h + \varepsilon + \delta h M. \tag{9}$$

The Bellman residual is zero here because the nominal tables are exact Bellman values. The broader model-error framework also allows a supplied table with a proved nonzero residual, but no residual term is silently introduced in these quote results.

Theorem 10.2 (Value and Q enclosures)**.** *For every compatible realization, every state, and every continuation horizon,*

$$|V_h(s) - \widehat{V}_h(s)| \le D_h, \tag{10}$$

$$|Q_h(s,a) - \widehat{Q}_h(s,a)| \le \rho_h(a) \le \Gamma_h. \tag{11}$$

Thus $L_h(s,a) = \widehat{Q}_h(s,a) - \rho_h(a)$ *and* $U_h(s,a) = \widehat{Q}_h(s,a) + \rho_h(a)$ *enclose the true Q value.*

Proof. At horizon zero the value error is zero. In a single backup, insert and subtract the expectation of the nominal continuation under the true row. The reward error contributes at most ε_a, the continuation-value error at most D_h, and row transport at most $\delta_a h M$. Taking a maximum over the same feasible action set is nonexpansive in the uniform score error. Therefore $D_{h+1} \le D_h + \varepsilon + \delta h M$, whose accumulated upper bound is (7). The local Q statement follows from the same inserted expectation before taking a maximum. □

The reconstructed argument follows `value_error`, `local_q_error`, `value_between`, and `q_between` in `AVOC/RobustQuotePlanning.lean`. The primitive steps are `reward_close`, `probability_l1_le`, and `expectation_transport`.

10.2 What the robust selector optimizes

The executable robust choice maximizes $L_h(s,a)$ over feasible actions. It is a maximizer of a *certified surrogate lower bound*. It is not asserted to solve the exact minimax problem over all compatible realizations. A bound can be valid and substantially below the actual worst-case score.

Proposition 10.3 (Distinct-competitor dominance)**.** *If a is feasible and*

$$U_h(s,b) \le L_h(s,a) \quad \textit{for every feasible } b \ne a,$$

then a is optimal in every compatible realization.

Proof. For a distinct competitor, chain the three inequalities $Q_h(s,b) \le U_h(s,b) \le L_h(s,a) \le Q_h(s,a)$. The comparison with a itself is reflexive. □

This is `competitor_dominance_sound` in `AVOC/CalibrationBridge.lean`. Excluding self-comparison is essential for a useful non-singleton certificate. The older sufficient condition compares every feasible upper score, including the chosen action's own upper score, to its lower score. That older condition is sound but forces the chosen radius to be zero. The formal declaration `self_comparison_forces_zero_radius` records this limitation.

Example 10.4 (A deliberately conservative abstract quote fixture)**.** The retained uncertainty example assigns zero setup to both kinds, warm gross 200, cold gross 60, warm success band $[3/5, 1]$, and exact cold success 1. Start with both statuses untried and use $h = 1$, hence two total actions. The nominal scores are warm 120, cold 60, and stop 0. Here $\varepsilon = 0$, $\delta = 4/5$, $M = 260$, and $D_1 = 0$. The local warm radius is 208, while the cold and stop radii are zero:

action	$\widehat{Q}_1$	ρ_1	L_1
inspectWarm	120	208	−88
generateCold	60	0	60
stop	0	0	0

The lower-Q rule chooses cold although warm's exact compatible Q range is $[120, 200]$. This is a clear example of conservatism in the certified envelope. It does not show that cold is the exact maximin choice.

The source proves the widths, local radii, and numerical order through `uncertain_warm_local_radius`, `uncertain_cold_local_radius`, and `uncertainty_changes_numerical_order`. The displayed first-action ranking is a direct reconstruction at the stated initial state and horizon. This fixture is *abstract quote planning*, not an artifact-backed saving example. Its protected rank-one source has dense cost 36 per call and the fixture has one call, so gross values 200 and 60 cannot come from bounded execution savings for that source. The separate 1000-call example above is the artifact-backed comparison. These two roles must remain distinct.

10.3 Whole-policy regret, not only first-action quality

Let $a_h(s)$ maximize $L_h(s,a)$ and let a^* maximize the true Q_h. Then

$$\begin{aligned} V_{h+1}(s) - Q_h(s,a_h(s)) &\le U_h(s,a^*) - L_h(s,a_h(s)) \\ &\le U_h(s,a^*) - L_h(s,a^*) \\ &\le 2\Gamma_h. \end{aligned}$$

The comparison controls the Bellman loss of the actual choice at each state. The policy is rebuilt at every remaining horizon with the same frozen evidence.

Theorem 10.5 (Expected frozen-policy regret)**.** *For a horizon H, let $J_H^{\text{lower}}(s)$ be the expected value of the complete adaptive policy induced by robust lower-Q selection. Under any compatible realization,*

$$0 \le V_H(s) - J_H^{\text{lower}}(s) \le \sum_{h=0}^{H-1} 2\Gamma_h.$$

The comparator is the true optimal legal adaptive plan with the same conservative reservation guards.

Proof. Write the regret at a node with $h+1$ remaining steps as its local Bellman loss plus the expected regret of the selected continuation. The local term is at most $2\Gamma_h$. By induction, every child regret is bounded by the common sum $\sum_{j<h} 2\Gamma_j$. Taking the expectation leaves that constant unchanged because probabilities sum to one. This yields the cumulative upper bound. Nonnegativity follows because the selected complete tree is legal and V_H dominates the value of every legal tree. □

This is the argument implemented by `plan_regret_le` in `AVOC/GuardedPolicyRegret.lean` and `robust_policy_regret` in `AVOC/RobustQuotePlanning.lean`. Ordinary finite sums simplify the displayed bound to

$$\varepsilon H(H+1) + \frac{\delta M}{3} H(H-1)(H+1).$$

This last closed form is a prose algebraic consequence of the source's sums, not a newly compiled Lean declaration. The envelope is deliberately broad and may be vacuous for a particular application. It remains a valid certificate. The regret is an expected policy-value difference. It does not bound every realized sequence of successes and failures.

11 Exact finite-grid calibration

Probability bands need an evidence rule. The implemented rule is an exact rational finite-grid cell-envelope construction. It should not be relabeled as an exact inversion of binomial tails or as a Clopper–Pearson interval. Its proof uses a different and simpler exclusion budget.

Let K be the number of successes in n Bernoulli trials with rational parameter $p \in [0,1]$. Define

$$b_{n,k}(p) = \binom{n}{k} p^k (1-p)^{n-k}, \qquad 0 \le k \le n.$$

Choose a positive integer grid size m and nonnegative rational allowance α. The grid cells have endpoints

$$\ell_i = \frac{i}{m}, \qquad u_i = \frac{i+1}{m}, \qquad 0 \le i < m.$$

For observed count k, the envelope of its point mass over cell i is

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

Retain the cell exactly when

$$B_{n,k,i} > \frac{\alpha}{n+1}.$$

Return the closed interval spanning all retained cells. If there are no retained cells, return $[0,1]$. The returned interval is therefore always a valid band. The construction performs exact rational arithmetic and strict comparison. Boundary cells cover 0 and 1 as well as interior parameters.

Theorem 11.1 (Finite-sample grid coverage)**.** *If K has the binomial count law $b_{n,k}(p)$, the returned interval $I(K)$ obeys*

$$\mathbb{P}\{p \notin I(K)\} \le \alpha.$$

This holds for every rational $p \in [0,1]$, without assuming that p is a grid point.

Proof. Some grid cell contains p. In that cell, $p \le u_i$ and $1-p \le 1-\ell_i$, hence $b_{n,k}(p) \le B_{n,k,i}$. If p is outside the returned interval, the containing cell could not have been retained: any retained cell lies inside the hull. Therefore $B_{n,k,i} \le \alpha/(n+1)$, and the mass of this excluded count k is at most that threshold. There are $n+1$ possible counts, so summing their failure masses gives at most α. If no cells were retained, the returned full band cannot exclude p and the conclusion is immediate. □

The source proves the grid cover, the cell envelope, excluded-count mass, and final sum as `grid_covers`, `mass_le_envelope`, `excluded_mass_le`, and `missMass_le` in `AVOC/FiniteCalibration.lean`. This proves finite-sample validity of the algorithm itself within the exact rational binomial model. It does not infer the sampling model from a list of reported outcomes.

Example 11.2 (Ten successes are evidence, not certainty)**.** For $n = 10$, $m = 10$, and $\alpha = 1/10$, ten successes retain cells $\{6,7,8,9\}$ and return $[3/5, 1]$. Ten failures retain $\{0,1,2,3\}$ and return $[0, 2/5]$. These endpoint calculations are explicit Lean theorems. By contrast, the empirical singleton $\{K/n\}$ has no such coverage property. At $n = 1$ and $p = 1/2$, both possible singletons exclude the true parameter, so its miss probability is 1.

The relevant declarations include `ten_successes_lower`, `ten_successes_upper`, `ten_failures_lower`, `ten_failures_upper`, and `empirical_singleton_counterexample`. Nothing here says the true parameter has posterior probability $9/10$ of lying inside an observed band. The guarantee concerns repeated-data failure probability under the specified law.

11.1 Predetermined schedules and data-dependent selection

A schedule is a finite set J of checkpoints. Each checkpoint fixes an identity stratum, sample size n_j, grid $m_j > 0$, true parameter p_j, and nonnegative error allocation α_j. Its count statistic has the stated binomial marginal. Counts can be dependent, including different prefixes of one source and checkpoints that share other underlying randomness.

Theorem 11.3 (Scheduled and selected failure bounds)**.** *Let B_j be the event that checkpoint j's interval excludes p_j. Then*

$$\mathbb{P}\Big(\bigcup_{j\in J} B_j\Big) \le \sum_{j\in J} \alpha_j.$$

For any data-dependent selector $j(\omega) \in J$,

$$\mathbb{P}\{\omega \in B_{j(\omega)}\} \le \sum_{j\in J} \alpha_j.$$

Proof. For each outcome ω, the indicator of a union is at most the sum of the individual indicators. Multiply by the nonnegative rational weight of ω and sum over the finite probability space. Interchange the two finite sums to obtain the union bound. Each checkpoint has failure mass at most its allocated α_j by finite-grid coverage. Finally, selected failure implies failure of at least one scheduled checkpoint, so it is a subset of the union. This last inclusion holds pointwise even if selection sees all outcomes. □

This reconstructs `finite_union_le`, `scheduled_binomial_failure_le`, and `selected_binomial_failure_le` in `AVOC/CalibrationSchedule.lean`. Neither checkpoint independence nor an independent selector is required. Predetermination is required: the theorem does not authorize creating arbitrary new checkpoints after viewing the data. It is finite scheduled selection, rather than an unrestricted anytime confidence sequence.

Example 11.4 (Why the schedule's total allowance is necessary)**.** Take a fair Boolean outcome ω and two checkpoint bad events $B_0 = \{\omega = 0\}$ and $B_1 = \{\omega = 1\}$. Each has probability $1/2$. Selecting $j(\omega) = \omega$ makes selected failure certain. The total allowance $1/2 + 1/2 = 1$ remains valid, while either single checkpoint's allowance fails. This is the exact obstruction proved by `fairBool_adaptive_exceeds_single_budget`.

12 From source laws to actual terminal samples

The count-law obligation is substantial. Fair marginals do not imply a binomial sum. Two identical fair bits each have parameter $1/2$, but their two-bit count is either zero or two and never one. The source contains this obstruction as `correlatedPair_not_product`.

The primitive model uses a finite rational probability space and, for each identity k, a finite latent Boolean vector of length N_k. Its success parameter p_k is fixed. The fresh-outcome contract states that after agreement with any specified pattern on a previously revealed set of positions, a fresh position's joint success mass is p_k times that history mass. In notation,

$$\mathbb{P}\{H \text{ matches and } X_i = 1\} = p_k \mathbb{P}\{H \text{ matches}\}, \qquad i \text{ fresh}.$$

The multiplication-only form is meaningful for histories of mass zero and does not divide by them. Induction determines the mass of every full pattern, then summing patterns by success count gives its binomial law. Truncation yields the same law for each fixed prefix.

The formal steps are `conditional_implies_product`, `count_fiber_eq_binomial`, and `prefix_count_fiber_eq_binomial` in `AVOC/ObservationStream.lean`. This derives the count law from a primitive source law. The primitive source law itself remains an explicit model assumption.

12.1 Pending intents and immutable terminal events

The new lifecycle recorder separates a pending-intent list from a terminal log. A single global natural-number operation-ID namespace covers both lists and all identities. Pending entries store an ID and identity. Terminal entries store an ID, identity, and success, rejection, or fault. An unfinished outcome cannot be inserted as a terminal event.

Command situation	Defined result
Fresh begin	Append one pending ID and identity.
Exact pending begin replay	Return the same state.
Begin replay after matching completion	Return the same state without reopening.
Matching fresh finish	Remove the pending ID and append one terminal event.
Exact terminal replay	Return the same state.
Changed identity or terminal outcome	Reject the command.
Finish without matching intent	Reject the command.
Unfinished terminal outcome	Reject the command.

These cases resolve a concrete inherited obstruction. If an immutable log stores an unfinished event under an ID, later adding success under that ID changes the event and must be rejected. A pending registry therefore cannot simply be an unfinished terminal record. The declaration `inherited_unfinished_terminal_conflict` proves the mismatch, rather than relying on a narrative explanation.

Theorem 12.1 (Pure recorder lifecycle invariant)**.** *Every successful run from the empty state has unique pending IDs, unique terminal IDs, disjoint pending and terminal IDs, and no unfinished terminal entries. An accepted terminal event occurs once and remains once throughout every later successful continuation. Terminal history grows by suffix append.*

Proof. For a begin step, the transition first checks both uses of the ID. It either returns the unchanged state for a matching replay, rejects conflicting reuse, or appends a fresh pending entry. For a finish step, exact replay is unchanged; otherwise successful completion requires an unused terminal ID and a matching pending entry. Removing that pending ID and appending its terminal event preserves uniqueness and disjointness. Unfinished outcomes are rejected before these cases. Induction over a successful command list preserves the invariant. Every successful step leaves the log unchanged or appends one event, which proves retention under continuation. □

The source declarations are `run_wellFormed`, `runFrom_log_prefix`, and `accepted_terminal_retained_once` in `AVOC/ObservationRecorder.lean`. A success adds true once to its identity's samples, a rejection adds false once, and a fault adds no sample. False outcomes count toward sample readiness just as true outcomes do.

12.2 A source-generative recorder and whole-run refinement

The raw recorder accepts externally supplied terminal labels. The generative recorder instead has begin, report, and fault commands. A fresh report constructs its true or false label from the next latent Boolean

of its identity. A replay checks the terminal log first and reuses the stored outcome, consuming no new source bit. A fresh report with an exhausted source rejects. Faults and begins consume no source bit. Reporting after a fault conflicts with the immutable terminal event.

Every accepted generative step emits an ordinary recorder command and an effective reveal marker. Whole-run induction proves that the emitted commands reproduce the actual raw recorder state and that the reveal state equals the inherited reveal machine on its effective trace. In particular, for every identity,

$$\mathrm{samples}_k(\mathrm{terminalLog}) = \mathrm{source}_k.\mathrm{take}\big(|\mathrm{samples}_k(\mathrm{terminalLog})|\big). \tag{12}$$

This is `run_refines` and `run_samples_eq_take` in `AVOC/RecorderReveal.lean`. The source-prefix correspondence is a derived invariant of the defined machine, not an additional equality asserted for each observed raw event. It still does not establish that an arbitrary external success label came from this latent source. A parser can check the label's syntax without checking its physical or statistical origin.

Example 12.2 (Replay and faults do not create new evidence)**.** Let an identity's source be [true, false]. Begin ID 7, report ID 7, then report ID 7 again. The sample list is [true] and the remaining source begins with false. Begin ID 8 and finish it as a fault. The sample list stays unchanged. Begin and report ID 9. It appends false, yielding [true, false]. A further fresh report rejects after source exhaustion, while a replay of ID 7 remains possible. The replay supplies no extra success and the fault supplies no concealed binary observation.

This is an illustrative execution of the defined transitions. The retained formal examples `replay_fault_exhaustion_example` and `exhausted_fresh_report_rejected` provide closely related checked instances. Reusing a different ID for the same physical operation is a separate identity problem. Global ID uniqueness alone cannot rule it out.

13 Unconditional calibrated policy issuance

The recorder-to-policy endpoint reads counts from the actual terminal log. Rejected whole runs supply no counts. Successful runs with *any* pending operation also supply no counts, even when the pending identity is unrelated to the selected checkpoint. A fully resolved run still needs enough success or rejection reports for both selected checkpoints. A checkpoint always reads the first predetermined n_j reports, so later reports cannot change that count.

When both selected counts are available, the endpoint constructs their exact learned probability bands and installs them for a fresh frozen policy episode. Setup and gross memberships are supplied deterministic premises. The warm and cold selectors may depend on all outcomes but must select checkpoints with the specified warm and cold true parameters. Changing a probability band does not authorize changing an outstanding operation's evidence.

Theorem 13.1 (Recorder-to-policy bad-issuance bound)**.** *Assume finite rational latent sources satisfy the primitive conditional laws; the finite schedule, grids, and nonnegative allowances are predetermined; scheduled prefix lengths fit within their sources; selected strata match the specified true probabilities; and deterministic setup/gross memberships hold. Then*

$$\mathbb{P}\{\textit{a plan is issued and lacks its episode certificate}\} \le \sum_{j \in J} \alpha_j. \tag{13}$$

The episode certificate supplies a compatible realization with the stated exact parameters, value and Q enclosures, and expected whole-policy regret for every finite horizon under frozen quotes.

Proof. The primitive conditional law yields binomial prefix count laws at all scheduled checkpoints. The generative run refinement gives (12). Whenever gated counts are available, taking the fixed first n_j reports therefore gives the corresponding latent prefix count. On the event where all scheduled bands cover their true parameters, either the gates suppress issuance or both selected learned bands contain the correct parameters. Combine those memberships with the supplied deterministic memberships to construct a compatible realization. The deterministic frozen-episode theorem then supplies the policy certificate. Thus bad issuance is a subset of the scheduled band failure union, whose probability is bounded by the total allowance. □

The final declarations are `recorder_episode_coverage` and `recorder_issued_failure_le` in `AVOC/RecorderPolicy.lean`. Success of execution, absence of pending intents, availability of enough samples, log-prefix equality, and the binomial count law are not separately supplied premises to that final theorem. They are derived or checked by its machine and gates. The primitive source laws and deterministic memberships are still supplied.

The probability in (13) ranges over the original outcome space, including rejected, unresolved, and insufficient-data runs. There is no claim that a plan will be issued, that any operation finishes, or that collection is cheap. If the issuance probability is $q > 0$, the generic conditional implication is only

$$\mathbb{P}\{\text{uncertified} \mid \text{issued}\} \le \min\left(1, \frac{\sum_j \alpha_j}{q}\right).$$

The same nominal allowance need not hold after conditioning on issuance.

13.1 Completion conditioning can select a biased population

Example 13.2 (Predictable availability still biases completed runs)**.** Let (X_1, X_2) be independent fair bits. Reveal X_1, and reveal X_2 by the deadline only if $X_1 = 1$. Availability of the second report depends on the past, not on X_2. Completion probability is $1/2$. Conditional on completion, $X_1 = 1$ with probability one, and the two-bit count has probability zero of being zero. A genuine two-trial binomial count with parameter $1/2$ has zero count probability $1/4$.

This is kernel-checked in `AVOC/ObservationCensoring.lean`, including `deadline_second_retention_predictable`, `deadline_conditioned_first_success`, and `deadline_conditioned_count_not_binomial`. It shows why one cannot replace the original law by the law conditioned on enough reports arriving.

A direct interval illustration uses ten independent fair bits and issues an interval only when all ten are true. Its issuance probability is 2^{-10}. The implemented $n = m = 10$, $\alpha = 1/10$ interval is then $[3/5, 1]$, which excludes $1/2$. Conditional on issuance, failure is certain. Unconditional bad issuance is only 2^{-10} and satisfies the allowance. This finite calculation is an illustrative consequence of the checked interval endpoints and the elementary product law, not a new Lean declaration.

13.2 Complete logs do not fix outcome-dependent completion order

Conditioning and ordering are different obstructions. Suppose two independent fair attempts start together. Success takes one unit of time and rejection takes two; resolve ties by the initiation ID. Record

every final outcome and wait until all intents have resolved. The completion order is

initiation outcomes	completion-order outcomes	mass
$(0,0)$	$(0,0)$	$1/4$
$(0,1)$	$(1,0)$	$1/4$
$(1,0)$	$(1,0)$	$1/4$
$(1,1)$	$(1,1)$	$1/4$

The first completed bit succeeds with probability $3/4$. No report was missing, no terminal event was forged, and no operation remains pending. Nevertheless, the first-completed prefix is not a fair source prefix.

For the implemented interval, the stronger retained finite experiment starts twenty independent fair attempts and lets all successes complete before all rejections. If there are at least ten successes, the first ten completed reports are all true. Their interval is $[3/5, 1]$ and excludes $1/2$. That event has mass

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

Thus actual interval failure exceeds its nominal allowance despite complete, fully resolved lifecycle records. The exact Python evidence is preserved under `results/v0.41.0/review_completion_order.py` and its JSON receipt. It is a finite computational obstruction with an explicit rational derivation here, not a newly kernel-checked theorem.

The generative theorem assigns the next latent bit in report order and assumes that source's conditional law. Independent outcomes in initiation order do not automatically satisfy that assumption after outcome-dependent reordering. A physical deployment must justify its report-order contract, for example through an initiation-indexed sampling policy or a proved alternative coupling. The current release leaves that physical coupling open.

14 Checked recovery and durable accounting

Recovery must distinguish a recorded intention from a completed paid result. The robust quote recovery model binds a commit to its sequence, complete pre-state, selected action, quote evidence, outcome tag, and post-state. Fresh acknowledgment checks those bindings and checks that the supplied typed outcome realizes the action through the runtime's `confirm` transition. It then requires the computed successor's store view to equal the recorded durable post-state. An unfinished intent is rejected rather than interpreted as a negative observation or permission to repeat paid work.

Proposition 14.1 (Fresh acknowledgment and replay)**.** *A successful fresh acknowledgment advances the sequence by one and yields the successor certified by `confirm`. It preserves the runtime safety invariant. Repeating that accepted commit at the resulting matching checkpoint returns the same checkpoint.*

Proof. Each failed identity, sequence, pre-state, outcome, evidence, or post-state comparison returns an error. A successful fresh branch therefore exposes all those equalities and a successful `confirm`. The successor invariant follows from `confirm_safe`. At the resulting checkpoint, the recorded last commit, next sequence, and actual store view match the duplicate branch, which returns the state unchanged. □

These are `acknowledge_fresh_contract`, `acknowledge_safe`, and `acknowledge_idempotent` in `AVOC/RobustQuoteRecovery.lean`. Duplicate acknowledgment recognizes the already accepted commit

and checkpoint. It does not freshly validate every field of an independently supplied outcome argument on that replay branch. This is appropriate for state replay but must not be described as new checking of a second physical result.

The `prepare` constructor selects the robust action and binds current evidence into its intent. Acknowledgment alone does not prove that an arbitrary supplied commit was originally produced by `prepare`. If the conclusion needs provenance of the planner's choice, that provenance must be supplied or established by the enclosing protocol. Fresh acknowledgment proves the exact checked transition contract, not an unstated origin claim.

Probability bands are frozen within the episode. Replacing them for a fresh episode preserves the source and protected identity fields but changes the approved evidence. An old intent cannot be acknowledged under the changed evidence. The explicit declaration `new_bands_obstruct_old_acknowledgement` in `AVOC/CalibrationBridge.lean` prevents calibration from silently rewriting an active durable operation.

14.1 Prepaid evaluation and upgrade journals

The separate durable-upgrade machine has a journal containing the current upgrade state, one optional pending ticket, and closed tickets. A ticket binds a campaign-local key, its complete parent state, and a proposal. Beginning a funded fresh ticket charges its protected evaluation tariff immediately. Repeating the same ticket is unchanged. Finishing it charges no second evaluation tariff, and switching occurs only according to the checked upgrade decision. Abandoning clears the pending ticket while retaining its already charged evaluation expenditure.

The upgrade authority supplies checker soundness and sound lower and upper utility bounds for one fixed workload, objective, and unit convention. These are explicit obligations in `AVOC/VerifiedUpgrade.lean`. Receiving a JSON utility bound or an approval flag does not prove them.

In particular, a failed or abandoned evaluation is not a free trial. The complete parent state includes spending, so reusing a ticket based on an old active version but ignoring changed expenditure is rejected. The formal declarations `begin_keeps_active`, `finish_evaluation_unchanged`, and `abandon_keeps_current` make these distinctions explicit.

For a successful paid trace with total evaluation debit e and switching debit u, the formal accounting identities are

$$\mathrm{evaluationSpent}_{\mathrm{final}} = \mathrm{evaluationSpent}_{\mathrm{initial}} + e, \tag{14}$$

$$\mathrm{switchSpent}_{\mathrm{final}} = \mathrm{switchSpent}_{\mathrm{initial}} + u. \tag{15}$$

From a safe initial journal, their sum stays within protected credits. Successful replay also cannot decrease the protected utility of the active version. These are `paid_trace_exact_accounting`, `paid_trace_no_overspend`, and `replay_utility_nondecreasing` in `AVOC/DurableUpgradeLaws.lean`.

Restoration replays supplied commands from a specified initial version and compares the result with the actual checkpoint. A successful restoration equals that checkpoint. Provided the specified initial version satisfies the authority's contract, restoration also inherits the safety certificate. This checks consistency between two supplied records. It cannot authenticate their physical origin or detect coordinated replacement of both the journal and checkpoint.

14.2 What recovery establishes about external execution

The lifecycle recorder is a pure finite snapshot reducer. Its once-only terminal history is not exactly-once external execution. It neither invokes an operation nor persists a crash-safe transaction. The recovery models give checked state-transition and replay contracts, while the retained runtime and storage adapters have their stated bounded evidence and external trust assumptions. Filesystem durability, endpoint authentication, physical identity, and reliable observation delivery are not consequences of a pure transition function.

A rejected pure batch supplies no final export. That is an all-or-nothing reduction result, not rollback of physical work already performed before rejection. A fault closes an intent without supplying a binary sample. It does not establish that retrying the corresponding paid operation is safe or free. Conservative reserved spending remains a protected model quantity rather than an automatic identity with actual billed expenditure.

The completed chain is therefore precise: under the declared finite source model, the lifecycle machinery derives trustworthy prefix correspondence; fixed scheduled calibration bounds bad issuance; on the simultaneous-coverage event, every issued frozen episode has the proved value and policy guarantees; outside that event, uncertified issuance has mass at most the scheduled total allowance; and the checked recovery machines preserve their specified state and accounting contracts. Physical source coupling, universal host/storage refinement, multi-episode accumulation, and measured productivity remain separate obligations.

15 Evidence, reproduction, and the publication boundary

The authoritative formal baseline for this edition is `AVOC_Formal_v0.41.0.zip`, with SHA-256

```
04efab87010b3b88c9a119275e190423a2465da32ba1e526edb7e6d2e0cd077e.
```

The original archive is preserved intact in the accompanying package. Its internal payload manifest was freshly checked for this edition: all 16,912 listed payload digests match. The archive contains the mathematical source, interfaces, experiments, review records, failed attempts, and reproduction instructions. It excludes the Lean runtime and compiled project/dependency objects.

15.1 Archived formal evidence

The v0.41.0 release record reports 222 positive build targets, all passing under warnings-as-errors, and 2,268 exact named theorem/lemma axiom reports. The allowed axiom union is `Classical.choice`, `Quot.sound`, and `propext`. There are 195 non-audit mathematical modules; the latest block adds four modules and 69 named theorem/lemma declarations. The 191 inherited mathematical modules were checked byte-identical to the preceding release. The archive also records 135 substantive negative proof rejections.

These are different counts with different meanings. A build target may be an executable or audit target rather than a mathematical module. A named theorem/lemma count includes local infrastructure and explicit private helpers. It is not a count of independent discoveries. Negative-test success means that the incorrect test proposition was rejected for a substantive proof reason, not merely that a command exited unsuccessfully.

The compiler pin is Lean 4.34.0-rc2, commit

```
6a10ac8c22beadecabdbb0919c2b50214762f91d.
```

The mathlib pin is

```
aeb72f1e9981b6d35b426b16dca7759be3e7ba06.
```

Nine exact dependency revisions are retained in `lake-manifest.json`. Local project objects were rebuilt from empty in the archived release. The external pinned caches were not rebuilt from every source. This distinction belongs to the reproducibility contract.

15.2 Fresh checks for this edition

A fresh source inventory matches the actual 195 mathematical modules and 2,268 named declarations to the delivered registry. It checks the archived declaration-source digests and recorded source locations. Fresh exact Python computations reproduce the declared examples and finite recorder comparisons. The detailed tested families, assertions, input digests, outputs, and commands are included in the publication's `reviews` directory.

No fresh Lean build or executable Lean comparison was performed for this publication edition. The compiler and exact portable dependency kit were unavailable in this session. The companion therefore uses archived kernel evidence together with fresh source-identity checks; it does not relabel that evidence as a new compilation. The Python comparisons remain bounded executable evidence and do not prove universal Python, parser, or storage refinement.

15.3 Assurance matrix

Layer	Established statement	Application obligation
Task context	Fiber criterion and decision-rule transport in the stated finite model	Identify the actual task and allowed rules
Artifact checker	Soundness/completeness for specified Boolean or affine syntax	Bind the checked input to the intended artifact
Cost comparison	Exact declared syntax/arithmetic tariffs and break-even calculations	Justify conversion to physical cost or reward
Frozen quote planner	Q/value enclosures, admissible robust choices, expected policy regret	True parameters and deterministic values belong to approved bands
Calibration	Finite rational cell-envelope coverage on fixed scheduled checkpoints	Source law, parameter strata, and predetermined schedule
Generative recorder	Unique logical IDs, immutable terminal retention, source-prefix refinement	Couple external operations and reporting order to the defined source
Recovery model	Confirmed logical evidence, selected installation, replay/accounting invariants	Input provenance and external execution/storage contracts

Layer	Established statement	Application obligation
Host programs	Exact bounded tests and archived Lean endpoint comparisons	Universal parsing/host refinement, authentication, physical persistence

15.4 The remaining research questions

The principal statistical question is the coupling from physical attempts to reported samples. In the generative model, a fresh accepted report consumes the next latent bit. In a physical service, an attempted outcome can affect latency, retention, fault reporting, or whether a report appears at all. The completion-order counterexample proves that merely resolving every intent does not solve this problem. An initiation-indexed sampling contract is a plausible future target, but this paper does not claim to have proved it.

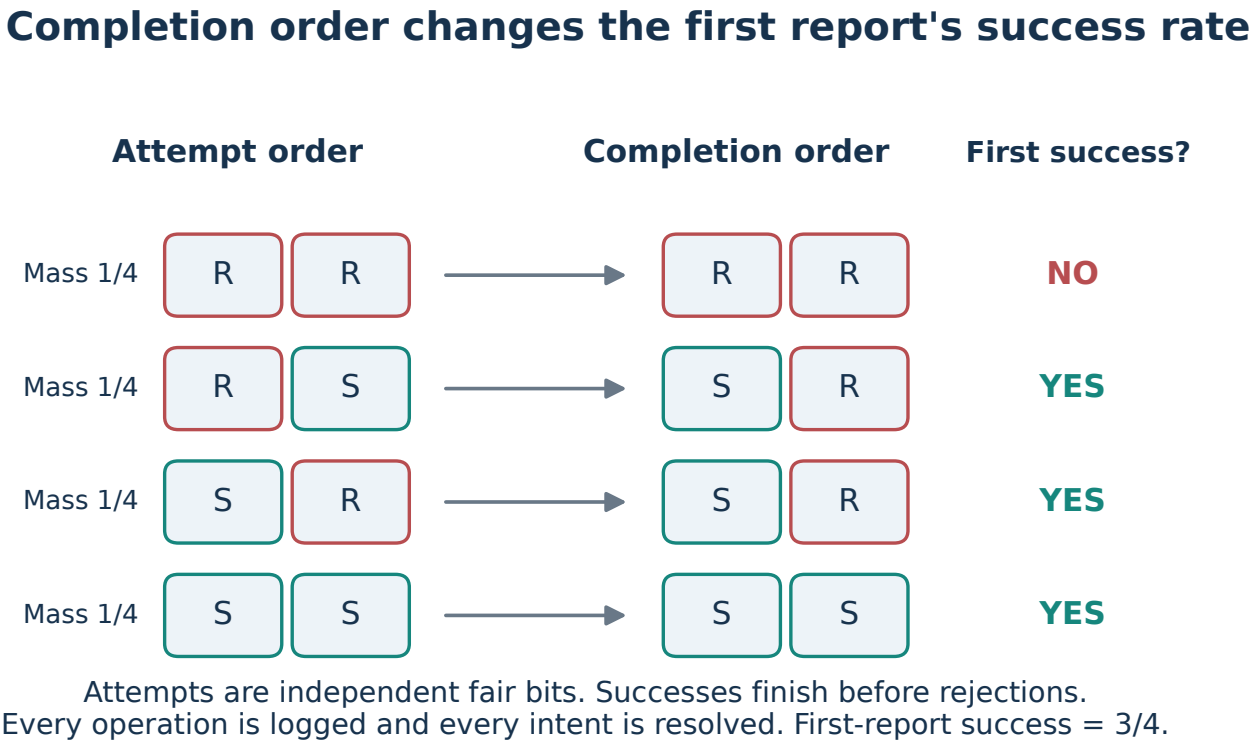


Figure 3: Every pair of attempted outcomes is logged and resolved, yet successes finishing first make the first reported bit successful with probability 3/4. This exact finite counterexample concerns a physical interpretation of reported samples; it does not refute the formal generative-source theorem.

A second question is controller composition. Current quotes are frozen within an episode. Updating bands during the same episode changes the policy and its dependency structure. Repeated episodes require explicit treatment of cumulative error, cost, and reward. The finite schedule theorem alone does not supply an unrestricted optional-stopping or continual-learning result.

A third question is implementation assurance. The raw recorder accepts supplied labels; it does not authenticate their truth or execute a paid operation. An injective identity registry can prevent protected

identity aliasing within its declared input contract, but a unique terminal ID does not prevent a caller from assigning two IDs to the same physical work. The inherited SQLite implementation relies on its database, operating-system, filesystem, and hardware contract. Its process-exit tests do not establish power-loss or distributed-system correctness.

Finally, the present source laws are finite and rational, and the artifact backends use explicit languages and tariffs. Broader real-valued probability models, drift, dependence, scalable calibration, unrestricted stopping, larger artifact languages, and measured service workloads remain distinct extensions. None is needed to state the existing formal results accurately.

16 Conclusion

AVOC supplies a checked finite route from artifact semantics and quoted uncertainty to admissible planning, calibrated policy issuance, and recoverable logical state. Its results are strongest when their interfaces are explicit. Correctness of an artifact, validity of a probability model, value of an action, and durability of external work are separate assertions, each with its own evidence.

The recorder composition closes an operational step inside the formal model: the samples used by calibration are the actual terminal samples of a complete generative run. The completion-order counterexample identifies the next external step with equal clarity. Together they provide a usable publication boundary: a substantial verified architecture, exact explanatory examples, and an explicit physical-sampling problem that remains open.

A Source and document navigation

The source root inside the original archive is `AVOC_Formal_v0.41.0`. Begin with `START_HERE.md`, `FORMAL_THEORY.md`, and `PROOF_BOUNDARY.md`. The full named declaration index is `THEOREM_INDEX.md`. Current reproduction instructions are in `REPRODUCE.md`; interface details are in `RECORDER_EXECUTABLE.md` and `OBSERVATION_EXECUTABLE.md`. Earlier source and verification records remain under `provenance`, `history`, and `results`.

Publication review maps identify which source statements support each manuscript section. The separate Lean companion gives the module groups, actual theorem names and excerpts, archived evidence schema, and practical reproduction commands. Figure programs and exact test outputs make the visual examples independently reproducible. These navigation aids supplement the original source rather than replacing its hypotheses.

References

[1] Komorebi P. *AVOC Formal v0.41.0*. Cumulative Lean/Python source and verification archive, 19 September 2026. Source, theory boundaries, declaration index, exact experiments, review records, and pinned reproduction instructions supplied with this edition.

[2] David Blackwell. *Comparison of Experiments*. In Jerzy Neyman (editor), Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, University of California Press, 1951, pp. 93–102. `https://digicoll.lib.berkeley.edu/record/112749`. Historical context for comparison of observations; the finite rule-transport proof in this paper is self-contained.

[3] The Lean development team. *Theorem Proving in Lean 4*, chapter "Axioms and Computation." https://lean-lang.org/theorem_proving_in_lean4/Axioms-and-Computation/. Consulted 7 October 2026 for the distinction between foundational axioms, kernel reduction, and compiled computation. This documentation is not a replacement for the project's pinned compiler and audit receipts.

[4] SQLite development team. *Atomic Commit in SQLite*. https://www.sqlite.org/atomiccommit.html. Official description of transaction mechanisms and hardware/flush assumptions, consulted 7 October 2026. Cited only for the external storage contract.