

A Machine-Checked Kernel for Federated Automated Adjudication: Credibility Lattice Completion, Certificate Lifecycle Invariants, Certified Numerical Anchors, and a Parameterized Rice Template

PIVOT-01 Federation (machine-generated; no human author claimed;
AI-use disclosure attached as supplementary material)

October 11, 2026 (v2; supersedes the registration-style v1, identifier 2610.00183v1)

Abstract

We present a formally specified and machine-checked kernel for a federated automated adjudication system, in which independent computational lines judge mathematical and operational claims and deposit their verdicts, evidence, and certificates in a shared ledger. The kernel comprises five components. (i) A three-rung, three-track credibility lattice obtained as a completion of a legacy five-element poset: the completion is justified by an exhaustive enumeration over all $11^3 = 1331$ ordered triples and is fully re-formalized in Lean 4, where the equational lattice laws, the order axioms, and the extremality properties of join and meet are all proved by the `decide` procedure over the finite carrier. (ii) A five-state certificate lifecycle machine with nine legal transitions and machine-checked invariants, likewise re-formalized in Lean 4; the formalization caught a specification flaw in one invariant (the originally stated grant-gating condition is false), and we report the corrected, machine-proved form. (iii) Certified numerical anchors: uniqueness of the gauged fixed point of circulant entropic-optimal-transport (Sinkhorn) systems, established by a Krawczyk interval operator with outward rounding across a six-instance family ($k \in \{6, 10\}$, $\varepsilon \in \{1, 1/2, 1/5\}$, later extended to ten instances), together with a second anchor family of five auction-algorithm assignment instances certified by interval dual brackets. We state plainly that the certification of these numerical anchors is carried by the Python interval-arithmetic layer; the attempted port to the Lean 4 interval library *lean-cert* is blocked by structural capability boundaries of that library (absence of the logarithm from its supported expression grammar and infeasibility of its determinant computation at the required dimensions), and the Lean files are formal statements, not machine-verified certificates. (iv) A parameterized Lean 4 template for Rice-type undecidability, consisting of a bridge theorem, an extensionality lemma, an instantiation generator, and discharged instances of increasing difficulty. (v) A minimal trusted computing base result: the soundness of an interval certificate checker (accept implies correctness), itself machine-proved in Lean 4. All Lean artifacts were verified on cloud infrastructure (Axiom Axle, environment `lean-4.28.0`) under the strict `verify_proof` channel, with axiom audits restricted to `propext`, `Classical.choice`, and `Quot.sound`, and no `sorryAx`. Empirical fuzzing over three certificate families observed zero false accepts; we report rule-of-three 95% upper bounds on the false-accept rate and explicitly do not claim a guarantee.

Contents

1	Introduction	2
1.1	The problem: adjudication among untrusted computational lines	2
1.2	What this paper deposits	3
1.3	Contributions	4

2	Related Work	4
3	Credibility Lattice Completion	5
3.1	The legacy poset and its gaps	5
3.2	The completion	6
3.3	Lean formalization	6
4	Certificate Lifecycle Machine	7
4.1	States and transitions	7
4.2	Invariants, including the one formalization broke	7
5	Certified Numerical Anchors	8
5.1	Circulant Sinkhorn fixed points: the Krawczyk family	8
5.2	Auction-algorithm assignment: the dual-bracket family	9
5.3	The boundary of the Lean port (FINDING-04)	9
6	A Parameterized Rice Template	10
6.1	Bridge, extensionality, generator	10
6.2	Discharged instances	10
7	Minimal Trusted Base	11
8	Reproducibility and Axiom Audit	11
8.1	Pinned artifacts and verification channels	11
8.2	Axiom audit	12
8.3	Environment fingerprint and dependency-drift caveat	12
8.4	Empirical soundness: fuzzing with rule-of-three bounds	12
8.5	Evidence-anchor table	13
9	Limitations and Honest Gaps	13
10	Conclusion	13
A	Pinned Artifact Index	14
B	Cloud Verification Request Identifiers	14

1 Introduction

1.1 The problem: adjudication among untrusted computational lines

Consider a federation of autonomous computational agents—“lines” in the terminology of our federation—that make claims about mathematical and operational facts: that a nonlinear system has a unique solution in a box, that a discrete optimization heuristic came within a certified bracket of the optimum, that a proposed inference rule preserves truth. The lines do not trust one another, and there is no central authority whose judgment is taken on faith. Any claim that is to be used downstream must therefore carry *machine-checkable evidence*, and the rules by which evidence is credited, promoted, maintained, downgraded, and revoked must themselves be small enough to audit and rigid enough to be machine-checked.

This is the federated automated adjudication problem. It is not primarily a problem of proving one hard theorem; it is a problem of *governance infrastructure for proofs*: a credibility order over claims, a lifecycle over certificates, numerical anchors whose acceptance cannot be fabricated by construction, a formal explanation of why fully automatic adjudication over unrestricted semantic properties is impossible, and a minimal trusted base in which the accept-implies-correct pattern is proved rather than assumed.

A naive design would adjudicate every semantic property of every program or certificate. Rice’s theorem [5] forbids this: any nontrivial extensional property of partial recursive functions is undecidable. The federation therefore adopts a *domain-limited, three-valued* architecture: adjudication objects are quadruples $o = (\varphi, D, \pi, V)$ consisting of a proposition φ , an explicit applicability domain D , a certificate π , and a checker V ; the adjudication map J is total by construction, taking values in $\{\text{pass}, \text{fail}, \text{undecided}\}$, with non-convergence within a round assigned the third value. Within this architecture, the soundness interface is the single clause

$$V(\varphi, D, \pi) = \text{accept} \implies \varphi \text{ holds on } D,$$

backed at each layer by a classical theorem: outward-rounding inclusion for interval evaluation, the Krawczyk theorem for existence and uniqueness, and LP weak duality for optimality brackets.

1.2 What this paper deposits

This paper is the full paper-form account of the kernel previously deposited in registration form (v1, identifier 2610.00183v1). Every component below is accompanied by machine certificates that are pinned, by content and by commit identifier, in the federation’s public repository, and every Lean 4 artifact carries a permanent cloud-verification request identifier under which the verification can be re-queried. The five components are:

1. **Credibility lattice completion** (Section 3). The legacy five-element credibility poset has exactly seven join gaps, found by enumeration; its completion is an eleven-element structure (three rungs $\times$ three tracks, plus formal top and bottom). The lattice laws are verified exhaustively over all $11^3 = 1331$ ordered triples, and the entire structure is re-formalized in Lean 4 with every law decided over the finite carrier (certificate CERT-LATTICE-01 / CERT-LATTICE-LEAN-01).
2. **Certificate lifecycle machine** (Section 4). A five-state machine (*candidate, granted, maintained, demoted, revoked*) with exactly nine legal and eleven illegal distinct transitions, with machine-checked gating, terminality, and freezing invariants (CERT-K4-01 / CERT-K4-LEAN-01). The Lean formalization refuted our original stronger statement of the grant-gating invariant; the corrected invariant is the machine-proved one.
3. **Certified numerical anchors** (Section 5). A six-instance (later ten-instance) family of gauged circulant Sinkhorn fixed-point systems, each certified to contain exactly one real root by a Krawczyk interval operator with outward rounding and negative controls; and a five-instance family of auction-algorithm assignment problems certified by interval dual brackets (CERT-CIRC-01 family, CERT-AUCTION-01). The certification carrier is the Python interval layer; we report, as a first-class result of the project, the structural boundary that blocks the Lean-side port (FINDING-04).
4. **Parameterized Rice template** (Section 6). A Lean 4 template that reduces any nontrivial extensional predicate on codes to Mathlib’s formalized Rice theorem, together with an

instantiation generator and seven discharged instances of increasing witness difficulty (T2A-BRIDGE-01, CERT-T2A-TEMPLATE-01).

5. **Minimal trusted base** (Section 7). A Lean 4 proof that the interval certificate checker is sound—acceptance implies the semantic property—for the step algebra used by the numerical anchors (CERT-SELF-CHECK-01).

1.3 Contributions

The contributions of this paper are infrastructural and methodological rather than a new theorem of independent depth:

- a complete, pinned, and machine-checked specification of an adjudication kernel, with every enumeration certificate reproducible from pinned scripts and every Lean artifact carrying a cloud-verification request identifier;
- a worked example in which formalization *caught* a specification flaw in a stated invariant (Section 4), which we regard as the strongest argument for the kernel’s methodology;
- an honest account of a failed port: the circulant family is certified at the Python interval layer, and its Lean-side formalization is structurally unverifiable under the pinned version of the target library; we document the exact boundary (Section 5.3);
- a reproducibility and axiom-audit protocol (Section 8), including rule-of-three honesty bounds on all empirical zero-failure claims.

Provenance and authorship stance. This paper is machine-generated by the PIVOT-01 federation orchestrator as part of an explicit experiment in machine-checked adjudication infrastructure. No human author is claimed; the contributing account holder does not claim human understanding of every derivation step. Good-faith correctness belief rests on the exhaustiveness of the enumeration certificates, the outward-rounding discipline of the interval certificates, kernel-level and cloud-level verification of the Lean artifacts, and the rule-of-three statistical framing of all empirical claims. An AI-use disclosure accompanies the submission as supplementary material, and the derivation transcripts are retained in the federation ledger.

Organization. Section 2 situates the work. Sections 3 through 7 present the five kernel components. Section 8 gives the reproducibility and axiom-audit protocol and the evidence-anchor table. Section 9 states limitations and honest gaps. Section 10 concludes. Appendix A collects the pinned artifact index.

2 Related Work

Proof assistants and Mathlib. Our formalizations are written in Lean 4 [1] and build on Mathlib [2], whose library of formalized mathematics includes a development of computability theory—partial recursive functions, codes, and Rice’s theorem in the form `ComputablePred.rice_2`—due to Carneiro [3]. Our Rice template (Section 6) is a thin, reusable layer on top of that foundation: the hard direction is entirely Mathlib’s, and our contribution is the parameterization and the instance generator. Our trust posture toward the proof assistant itself follows the spirit of work on the metatheory and self-verification of Lean, such as Lean4Lean [4]; we do not rely on such results, and instead minimize and audit what our artifacts assume (Section 8).

Interval arithmetic and Krawczyk certification. Interval arithmetic as a tool for rigorous numerical bounds goes back to Moore [6]. The Krawczyk operator [7] converts a candidate root and a box into a certified uniqueness statement: if the operator maps the box strictly into itself, the box contains exactly one root. Our numerical anchors (Section 5) use exactly this criterion, with rationalized centers, rationalized analytic-inverse preconditioners, and outward rounding at every step, so that the acceptance of a certificate is a theorem about real numbers rather than about floating point. The accept-implies-correct shape of these certificates is what our minimal trusted base (Section 7) formalizes in miniature.

Optimal transport and auction algorithms. Entropic regularization of optimal transport and its Sinkhorn iteration are classical [8, 9]. We do not certify the iteration; we certify the *fixed-point system* directly for circulant cost matrices, where a closed-form solution exists and serves as ground truth for negative controls. The auction algorithm for assignment problems is due to Bertsekas [10]; our second anchor family certifies auction output a posteriori via interval dual brackets (LP weak duality), with the solver treated as an untrusted candidate generator.

Undecidability and the shape of the escape. Rice’s theorem [5] is the formal reason that the federation’s adjudication map is three-valued and domain-limited. We treat the theorem not as background but as a component of the kernel: the template of Section 6 makes the blocking result callable, so that any proposed extension of the adjudication domain to a new semantic property must first pass (or be blocked by) a discharged instance.

Registration platforms for machine-generated mathematics. This submission is itself an experiment in the registration of machine-generated, machine-checked mathematical content: the authorship mode is *no-human-author-claimed*, all artifacts are pinned in a public repository, and the derivation transcripts are retained. A predecessor of this paper was desk-rejected in moderation as “not a paper”; the present version is a deliberate test of whether a fully paper-form treatment of the same evidence base meets the venue’s expectations, and the outcome of that test is itself data for the federation’s platform-adaptation ledger.

Statistical honesty for zero-failure observations. When n independent trials observe zero failures, the rule of three [11] gives $3/n$ as an approximate 95% upper confidence bound on the failure rate. All empirical zero-false-accept claims in this paper are stated with such bounds and with the explicit wording “no false accept observed”, never as guarantees.

3 Credibility Lattice Completion

3.1 The legacy poset and its gaps

The federation’s credibility grades descend from a legacy five-element poset: three rungs of evidential strength (*candidate* < *empirical* < *domain-formal*) plus two cross-cutting grades (*mirror-insight* for insight-track claims and *policy* for governance-track claims). Grades are used to gate which claims may serve as premises in deduction, so the poset’s *join* operation is load-bearing: the join of the grades of two premises is the natural grade of any conclusion drawn from both.

An exhaustive enumeration over all pairs of the five-element poset (certificate CERT-LATTICE-01) found that the poset has *exactly seven join gaps*: pairs with no least upper bound. The gaps are exactly the cross-track pairs: each of the three rungs fails to join with *mirror-insight* and with

policy (six pairs), and *mirror-insight* fails to join with *policy* (the seventh). A poset with join gaps cannot support premise combination; completion is forced.

3.2 The completion

Definition 3.1 (Credibility lattice). The carrier is $E = (R \times T) \cup \{\top, \perp\}$, where $R = \{\text{candidate} < \text{empirical} < \text{domain-formal}\}$ is the three-rung chain and $T = \{\text{adjudication}, \text{insight}, \text{governance}\}$ is the set of tracks. The order compares rungs within a track; $\perp$ is below everything and $\top$ above everything; cells of different tracks are incomparable. Join of two cells is the rung-maximum within a common track and $\top$ across tracks; meet is the rung-minimum within a common track and $\perp$ across tracks.

Remark 3.2. The completion embeds the legacy poset order-preservingly and cleanly: *candidate*, *empirical*, *domain-formal* map to the adjudication track at their rungs, *mirror-insight* maps to (candidate, insight), and *policy* maps to (candidate, governance). The enumeration reports order-preservation true and reflection violations none (CERT-LATTICE-01).

Theorem 3.3 (CERT-LATTICE-01: exhaustive lattice verification). *Commutativity of join and meet, associativity of join and meet, and both absorption laws hold for all $11^3 = 1331$ ordered triples of the eleven-element structure of Definition 3.1 (exhaustive machine check, zero failures).*

3.3 Lean formalization

The full structure—carrier as an inductive type, order, join, meet—is formalized in Lean 4 (`artifacts/lean-verification/certificate CERT-LATTICE-LEAN-01`). Because the carrier is finite (eleven elements), every universal statement over it is decidable, and every theorem below is proved by the single tactic `decide`, which reduces to kernel evaluation; cloud verification request `1dfa70b6-ba1e-492d-b45c-5391727aff3c` (Axiom Axle, `lean-4.28.0`, `okay = true`).

Theorem 3.4 (CERT-LATTICE-LEAN-01). *Over the carrier of Definition 3.1, all of the following hold and are machine-decided in Lean 4:*

1. *the six equational lattice laws: `join_comm`, `meet_comm`, `join_assoc`, `meet_assoc`, and the two absorption laws `absorb1`, `absorb2`;*
2. *the order axioms: `le_refl`, `le_trans`, `le_antisymm`;*
3. *extremality: `join_is_lub` (the join is below every common upper bound) and `meet_is_glb` (the meet is above every common lower bound);*
4. *order-operation compatibilities: `le_join_left`, `le_join_right`, `meet_le_left`, `meet_le_right`.*

Remark 3.5 (Declaration count). The certificate header describes a fourteen-theorem core package; the pinned file contains fifteen `decide`-proved declarations in total, the fifteenth being the fourth order-operation compatibility listed above. We report the file’s actual content.

The axiom audit (Section 8) confirms that these theorems depend only on `propext`, `Classical.choice`, and `Quot.sound`—the standard Lean 4 axiom whitelist—with no `sorryAx`. Because the proofs are by `decide`, their content is identical to the Python enumeration of Theorem 3.3; the value of the Lean layer is that the *statement* is machine-checked for well-formedness against a single source of truth, eliminating transcription drift between the specification and the enumeration script.

4 Certificate Lifecycle Machine

4.1 States and transitions

Certificates in the federation are not static. A deposited certificate is born as a *candidate*, may be *granted* after a four-gate review, is *maintained* under continuous monitoring, may be *demoted* when a domain violation is detected, and may be *revoked* when forgery is detected or its domain is withdrawn.

Definition 4.1 (Lifecycle machine). The state set is $S = \{\text{candidate}, \text{granted}, \text{maintained}, \text{demoted}, \text{revoked}\}$. The legality predicate $\ell : S \times S \rightarrow \{\text{true}, \text{false}\}$ admits exactly the following nine transitions:

From	To	Trigger
candidate	granted	four-gate review
candidate	revoked	domain withdrawal
granted	maintained	continuous monitoring passed
granted	demoted	domain violation detected
granted	revoked	certificate forgery detected
maintained	demoted	domain violation detected
maintained	revoked	certificate forgery detected
demoted	granted	remediation re-review
demoted	revoked	forgery detected or domain withdrawal

4.2 Invariants, including the one formalization broke

Theorem 4.2 (CERT-K4-01 / CERT-K4-LEAN-01). *The machine of Definition 4.1 satisfies, all by exhaustive machine check and all re-proved in Lean 4 by `decide` over the five-state carrier (cloud verification request 16618831-e457-4d29-8d65-b57fa563e33d):*

1. *legal_count*: exactly 9 transitions are legal;
2. *illegal_count*: exactly 11 distinct (non-self) transitions are illegal;
3. *I1_granted_gated* (grant gating): if $\ell(s, \text{granted})$ then $s \in \{\text{candidate}, \text{demoted}\}$;
4. *I1b_maintained_only_from_granted*: if $\ell(s, \text{maintained})$ then $s = \text{granted}$;
5. *I2_revoked_terminal* (revocation terminality): no transition leaves revoked;
6. *I2b_no_return_to_candidate* (irreversibility): no transition returns to candidate;
7. *I3_demoted_out_edges* (appeal freezing): if $\ell(\text{demoted}, s)$ then $s \in \{\text{granted}, \text{revoked}\}$;
8. *no_self_loop*: $\ell(s, s)$ is false for every s .

Remark 4.3 (A specification flaw caught by formalization). Our original specification stated invariant (I1) more strongly: “*granted* arises only from *candidate*”. This is *false*: the remediation edge $\text{demoted} \rightarrow \text{granted}$ is a legal entry into *granted*. The Lean formalization’s `decide` procedure produced a counterexample to the original statement, and the invariant was corrected to the disjunctive form in Theorem 4.2(3), which is the machine-proved one. We regard this incident as the methodological point of the exercise: the informal specification, reviewed by humans and by machines at the enumeration layer, had silently dropped an edge from the gating invariant; only the act

of stating the invariant in a proof assistant exposed the gap. The corrected statement is recorded as the “Lean-ification precision correction” in the certificate header of `CertK4Lean01.lean`.

The Python enumeration layer (CERT-K4-01, pinned with raw data in the federation ledger) independently confirms the counts and the three invariants over the full transition graph, including the enumeration of all eight candidate-originating paths for the append-only evidence property and the closure of all five states’ outgoing edges under a raised appeal flag.

5 Certified Numerical Anchors

The kernel’s adjudication machinery would be vacuous without at least one class of certificates whose acceptance is anchored in mathematics rather than in trust. This section presents two such anchor families and then states, as a first-class finding, the boundary of the Lean-side port.

5.1 Circulant Sinkhorn fixed points: the Krawczyk family

For a circulant cost matrix $C[i][j] = c_{(j-i) \bmod k}$ with $c = (c_0, \dots, c_{k-1}) \in \mathbb{Q}_{>0}^k$, uniform marginals, regularization $\varepsilon > 0$, and the gauge $f_0 = 0$, the entropic optimal-transport (Sinkhorn) fixed-point system reads

$$f_i + \varepsilon \log k + \varepsilon \log \sum_j e^{(g_j - C[i][j])/\varepsilon} = 0, \quad g_j + \varepsilon \log k + \varepsilon \log \sum_i e^{(f_i - C[i][j])/\varepsilon} = 0.$$

Circulant symmetry forces the closed-form solution

$$f^* = 0, \quad g^* = -\varepsilon (\log k + \text{lse}(-c/\varepsilon)),$$

where lse is the log-sum-exp; e.g. for the $k = 6$, $\varepsilon = 1$ instance with $c = (1/2, 1/3, 2/3, 1/4, 3/4, 1/5)$, $f^* = 0$ in the gauge and the common g -value is $-3.1543698608222522\dots$, reproduced digit-for-digit by the independent batch verification. The closed form is used only as ground truth for negative controls; the certificate does not consult it.

The certificate is the classical Krawczyk criterion [7]. With F the $(2k - 1)$ -dimensional gauged system, $\hat{x}$ a rationalized center (rational approximation of the closed form, `limit_denominator` at precision 10^{-13} for the original batch and $\sim 10^{-26}$ for the re-verification batch), C a rationalization of the analytic inverse Jacobian $J(\hat{x})^{-1}$, and X the box of radius 10^{-9} around $\hat{x}$, the operator

$$K(X) = \hat{x} - C F(\hat{x}) + (I - C J(X))(X - \hat{x})$$

is evaluated in interval arithmetic with outward rounding (mpmath interval layer, 80 decimal digits). If $K(X)$ lies strictly inside X and C is nonsingular, X contains exactly one real root.

Theorem 5.1 (CERT-CIRC-01 family; Python interval certification). *For each of the six instances $k \in \{6, 10\}$, $\varepsilon \in \{1, \frac{1}{2}, \frac{1}{5}\}$ (two circulant vector families, dimensions $n = 11$ and $n = 19$), the Krawczyk operator maps the box strictly inside itself: Jacobian condition numbers 22.2–43.1, Krawczyk image widths between $2.55 \cdot 10^{-15}$ and $6.93 \cdot 10^{-14}$, against a box radius of 10^{-9} (margin of more than four orders of magnitude). Hence each box contains exactly one real root. Every negative control—the same box with centers perturbed by 10^{-6} , preconditioner recomputed—is rejected. An independent re-verification batch (EXT05-BATCH-01) reproduced all six instances (condition numbers matching to published precision) and extended the family to $\varepsilon \in \{1/10, 1/20\}$, for a total of 10/10 instances with strict inclusion and 10/10 negative controls correctly rejected; at $\varepsilon = 1/20$ the condition number rises to $1.1 \cdot 10^3$ ($k = 6$) and the image width stays below $5 \cdot 10^{-14}$.*

Remark 5.2. The extension to small ε is where the certification earns its keep: the closed form is cheap, but the *box uniqueness* statement is what downstream adjudication consumes, and it survives condition numbers about one and a half orders of magnitude above the original family (to $\approx 1.1 \cdot 10^3$).

5.2 Auction-algorithm assignment: the dual-bracket family

The second anchor family (CERT-AUCTION-01) certifies the output of the auction algorithm [10] on assignment problems with log-uniform costs $C = 10^{R(2U-1)}$, U uniform on $[0, 1]^{k \times k}$, dynamic range $R = 2$. The solver (forward auction with ε -scaling, $\varepsilon : 0.5 \rightarrow /5 \rightarrow 10^{-6}$ per round, warm prices) is treated as an untrusted candidate generator; the certificate is a posteriori and rests only on LP weak duality: an interval evaluation (outward rounding via directed rounding of endpoints) certifies reduced-cost nonnegativity $\text{rc}_j = c_j - (A^T y)_j \geq 0$ componentwise, with a δ -deflation ($\delta = 10^{-10}$) at degenerate zero-riding coordinates; the assignment-scale bracket is $k \times [\text{LB}_{\text{lo}}, \text{UB}_{\text{hi}}]$ in the Birkhoff scaling.

Proposition 5.3 (CERT-AUCTION-01; Python interval certification). *All five instances (one benchmark at $k = 8$, $R = 2$, seed 7; three extended instances at $k \in \{6, 10, 12\}$; one schedule-variant re-run of the benchmark) pass the three-part test: the auction cost lies in the certified bracket, the gap satisfies $\text{cost} - k \cdot \text{LB}_{\text{lo}} \leq k \varepsilon_{\text{final}}$, and a swap-perturbation negative control (perturbed cost 97.28–130.71 against brackets of width $\sim 5 \cdot 10^{-11}$) falls outside the bracket and is rejected. The benchmark reproduces the originally registered cost 0.2550220110027003 digit-for-digit, and the schedule-variant converges to the same assignment, family-level evidence of path independence of ε -scaling.*

Remark 5.4 (Honesty boundary of the auction family). The upper side of the bracket uses the LP solver’s primal solution as the candidate; primal feasibility is trusted to solver precision ($\sim 10^{-12}$) and is *not* interval-certified—this is the original specification, and we flag it rather than repair it silently. The family has no Lean port.

5.3 The boundary of the Lean port (FINDING-04)

We attempted to port the six circulant instances to Lean 4 using the verified interval library *leancert*, encoding each system over *leancert*’s expression grammar and calling `verify_unique_system_root` with `native_decide`. The port does not go through, and the failure is structural, not incidental:

1. *leancert*’s supported-expression check (`ADSupported / checkADSupported`) does not include `Expr.log`—it never has, since the commit that introduced the function—while all 11/11 (resp. 19/19) equations of every circulant system contain `log`. The checker therefore rejects by design: probes measure `checkADSupported = false` on 11/11 equations while `centerInside = true`.
2. *leancert*’s determinant (`Matrix.det`) is defined by Leibniz expansion; at dimensions 11 and 19 the computation is combinatorially infeasible for `native_decide` (the library’s own tests cover only 2×2).

Consequently, under the artifact-pinned *leancert* v4.34.1 the circulant Lean family *has never passed* Lean-side machine verification: after elaboration-layer repair the files elaborate cleanly (ELABORATION OK), but `native_decide` evaluates to false for the structural reasons above, and the repaired artifacts carry the honest verdict `CHECK_FALSE (STRUCTURAL)`. **The certification of the circulant family is carried entirely by the Python interval layer** (outward rounding, strict-inclusion criterion, nonsingular preconditioner, negative controls), and the Lean files are *formal*

statement artifacts—machine-checked transcriptions of the systems, centers, boxes, and claims—not machine-verified certificates. A content-level repair path is registered as an open obligation: eliminate the logarithm by the substitution $u = \log \sum \exp(\cdot)$ to bring the systems into the library’s supported fragment, or migrate to a checker with transcendental support; the rationalized centers for the extended instances are already registered for this purpose.

We document this boundary in the body of the paper, rather than in a footnote, because it is exactly the kind of fact that an adjudication kernel exists to protect: a credential whose presentation outruns its certification layer. The v1 submission’s wording (“ported to the Lean 4 verified interval library *leancert*”, abstract; “`verify_unique_system_root` certificates”, §4) was, under this finding, overstated, and is corrected here.

6 A Parameterized Rice Template

6.1 Bridge, extensionality, generator

The domain-limitation architecture of Section 1 is justified by Rice’s theorem [5], and Mathlib contains a formalization of that theorem over partial-recursive-function codes [3]. The kernel’s fourth component (T2A-BRIDGE-01, CERT-T2A-TEMPLATE-01) packages the blocking result as a callable template in Lean 4. All six theorems below were cloud-verified under `verify_proof` (Axiom Axle, `lean-4.28.0`, `okay = true`, request identifiers in Table 2), with axiom audits clean.

Proposition 6.1 (Bridge, `rice_bridge`). *Let $C \subseteq \text{Code}$ be extensionally closed (codes with equal evaluation semantics are equimembership) and nontrivial ($C \neq \emptyset$, $C \neq \text{univ}$). Then membership in C is not a computable predicate.*

The proof is a three-line reduction to Mathlib’s `ComputablePred.rice_2`: decidability of C would force $C = \emptyset$ or $C = \text{univ}$.

Lemma 6.2 (Extensionality of pointwise predicates, `ext_of_pointwise`). *For any $g : \mathbb{N} \rightarrow \mathbb{N}$, the predicate “ $\exists m, \text{eval } c \ m = \text{some } (g \ m)$ ” is extensionally closed in the sense of Proposition 6.1.*

Theorem 6.3 (Generator, `rice_pointwise`). *For any $g : \mathbb{N} \rightarrow \mathbb{N}$ whose range avoids some point k (i.e. $\forall m, g \ m \neq k$), the pointwise predicate of Lemma 6.2 is undecidable.*

Proof sketch. Instantiate the bridge with $C = \{c \mid \exists m, \text{eval } c \ m = \text{some } (g \ m)\}$. Extensionality is Lemma 6.2. Nontriviality is witnessed by the constant functions: `const (g 0)` is in C (at $m = 0$), and `const k` is not (the range-avoidance hypothesis). The Lean proof is a direct formalization of this argument. $\square$

6.2 Discharged instances

The generator is discharged on seven instances of increasing witness difficulty (Table 1). The first three accompany the template file itself; the remaining four (INSTANCES-EXT01) demonstrate that the generator’s call surface extends from constant and linear witnesses to nonlinear arithmetic, inductive witnesses, and divisibility arguments. All seven carry cloud verification request identifiers and clean axiom audits; the EXT01 batch was additionally verified as a single four-theorem artifact (request `77a0eaad-fbd2-48a6-9281-330f736fc8c2`).

Remark 6.4 (Engineering note). The EXT01 batch recorded a reusable engineering fact: Lean’s `omega` tactic does not see through β -redexes, so instance proofs first reduce the goal with `show` before invoking `omega` or `rw`. This is the kind of artifact-level detail that the federation ledger retains and that a template’s reusability depends on.

Table 1: Discharged instances of the Rice template. “Avoids” is the point k of Theorem 6.3.

Instance	Semantic property (g)	Avoids	Witness content
<code>inst_const_zero</code>	output constantly 0	$k = 1$	<code>Nat.zero_ne_one</code>
<code>inst_succ</code>	output = input + 1	$k = 0$	<code>Nat.succ_ne_zero</code>
<code>inst_double</code>	output = $2m$	$k = 1$	parity (<code>omega</code>)
<code>inst_shift7</code> (E4)	output = input + 7	$k = 3$	β -redex elimination + <code>omega</code>
<code>inst_square_ne3</code> (E5)	output = m^2	$k = 3$	case split: $m < 2$ enumeration; $m \geq 2 \Rightarrow m^2 \geq 4$
<code>inst_pow2_ne3</code> (E6)	output = 2^m	$k = 3$	induction: $2^0 = 1$, 2^{m+1} even
<code>inst_fact_ne4</code> (E7)	output = $m!$	$k = 4$	divisibility: $m \geq 3 \Rightarrow 6 \mid m!$

7 Minimal Trusted Base

Every anchor certificate ultimately reduces to: a checker computed *accept*, therefore the semantic property holds. The kernel’s fifth component (CERT-SELF-CHECK-01) proves this pattern, in miniature, in Lean 4, over rational intervals.

Definition 7.1 ($\mathbb{Q}$ -intervals and the checker). An interval `iv` is a pair $(\text{lo}, \text{hi}) \in \mathbb{Q}^2$; membership `mem` is $\text{lo} \leq x \leq \text{hi}$. Interval addition and negation are endpointwise. The boundedness checker `check (iv, B)` is the proposition $-B \leq \text{lo} \wedge \text{hi} \leq B$.

Theorem 7.2 (CERT-SELF-CHECK-01: *accept* implies correctness). *The following are proved in Lean 4 (cloud verification under `verify_proof`, `lean-4.28.0`, `okay=true`; request identifiers in Table 2; axiom audit clean):*

1. *`add_sound`: interval addition preserves membership;*
2. *`neg_sound`: interval negation preserves membership;*
3. *`check_sound`: if `check (iv, B)` holds, then every $x \in \text{iv}$ satisfies $|x| \leq B$;*
4. *`end_to_end`: composition—if the certificate arithmetic `add a (neg b)` passes `check` against B , then $|x - y| \leq B$ for all $x \in a$, $y \in b$.*

Remark 7.3. This is deliberately tiny. Its role in the kernel is foundational rather than mathematical: it is a machine-checked instance of the soundness interface $V(\varphi, D, \pi) = \text{accept} \Rightarrow \varphi$, exhibiting that the interface is satisfiable by a real, verified artifact. The full Krawczyk-layer analogue—a verified interval evaluator with outward rounding strong enough to carry Theorem 5.1 inside Lean—is precisely what FINDING-04 (Section 5.3) shows to be beyond the pinned *leancert* version; it is registered as open obligation OBL-A1.

8 Reproducibility and Axiom Audit

8.1 Pinned artifacts and verification channels

All artifacts are pinned by path and commit in the public repository `chepin-ai/vci-inbox`; the index is in Appendix A. The verification channels are of three kinds:

- *Exhaustive enumeration* (Python, pinned scripts): lattice laws over 1331 triples; lifecycle transition counts and invariants. Raw data are frozen in the ledger (board/LAB-THEORY-01R-FK01R-20261009T0620Z.md, commit `3e0f54e1`; reproduction script commit `16ed41d2`).

- *Interval certification* (Python, mpmath interval layer, outward rounding, 80-digit precision): circulant Krawczyk family (EXT05-BATCH-01 script and machine-readable results under `artifacts/CERT-CIRC-LEAN-01/family/`) and the auction dual-bracket family (`artifacts/CERT-AUCTION-`
- *Cloud proof verification* (Axiom Axle, environment `lean-4.28.0`, strict `verify_proof` channel: no `sorry`, whitelisted axioms, signature match): all Lean artifacts of Sections 3–7. Each theorem carries a permanent request identifier under which the verification can be re-queried; the identifiers are listed in Table 2 and in the artifact headers.

8.2 Axiom audit

Every Lean artifact was audited in the cloud environment with `#print axioms`. For every artifact the recorded audit reports the dependency set as a subset of the standard Lean 4 whitelist at artifact level

`{propext, Classical.choice, Quot.sound},`

and `sorryAx` appears in none. Because the lattice and lifecycle theorems are proved by `decide`, their logical content reduces to kernel computation; the choice axiom enters only through Mathlib’s `Finset` machinery and is inessential to the checked statements.

8.3 Environment fingerprint and dependency-drift caveat

The cloud verification environment fingerprint is `lean-4.28.0` (Axiom Axle). The attempted *leancert* port additionally records a dependency-pinning incident relevant to reproducibility: the upstream tags `v4.34.1` / `v4.35.0` of *leancert* were observed to have been retargeted on the remote (the tag target differs from the artifact-pinned commit), so the artifact pins by commit hash and the drift risk is registered in the federation ledger. We reproduce this caveat because any consumer of the formal statement artifacts must resolve the dependency by the pinned commit, not by tag.

8.4 Empirical soundness: fuzzing with rule-of-three bounds

Three certificate families were fuzzed, 120 trials in total (certificate CERT-T4-01, raw data in the pinned ledger card; f_{80} ground truth means containment of the 80-bit-extended-precision value, judged bitwise):

- **E family** (interval envelopes): $n = 30$, ground-truth containment 30/30;
- **D family** (LP dual brackets, with implanted corrupted subsets): $n = 60$, accepts 4, all 4 valid;
- **K family** (Krawczyk, adversarial centers with implanted true center): $n = 30$, accept 1, independently verified by an f_{80} Newton iteration.

Zero false accepts were observed. The three layers are deliberately non-identically distributed and we do not extrapolate a universal statement. The exact one-sided 95% upper confidence bounds (Clopper–Pearson) on the false-accept rate are 9.5% (E), 4.9% (D), 9.5% (K); the rule-of-three $3/n$ approximation [11] gives 10%, 5%, 10%. **We report “no false accept observed”, not a guarantee.**

8.5 Evidence-anchor table

9 Limitations and Honest Gaps

We collect the limitations in one place, matching the federation’s honesty discipline that every claim be traceable to an artifact and every gap be named rather than elided.

1. **Machine generation and human comprehension.** This paper and all of its artifacts are machine-generated under the authorship mode *no-human-author-claimed*. The contributing account holder does not claim human understanding of every derivation step; the correctness belief rests on the verification layers enumerated in Section 8, not on human line-by-line review. Human comprehension of some derivation steps is therefore partial by design, and readers should weight the machine certificates accordingly—this cuts both ways: the certificates are stronger than a human read in exhaustiveness, and weaker in semantic judgment about what *should* have been proved.
2. **The circulant Lean boundary.** The circulant family’s certification is carried by the Python interval layer. The Lean-side files are formal statement artifacts that elaborate but cannot be machine-verified under the pinned *lean-cert* v4.34.1, for the two structural reasons of Section 5.3 (no `Expr.log` support; infeasible Leibniz determinant at $n \in \{11, 19\}$). Any v1-era wording implying a completed Lean-side certificate for this family was overstated and is corrected here. The repair path (log-elimination by substitution, or a checker with transcendental support) is an open obligation, not a result.
3. **The auction family’s trusted edge.** The dual bracket’s upper side trusts solver primal feasibility to $\sim 10^{-12}$; it is not interval-certified (Remark 5.4). The family has no Lean port.
4. **No statistical guarantees.** All fuzzing conclusions are sample-based, non-identically distributed across layers, and reported with rule-of-three upper bounds. “Zero false accepts observed” is the complete claim.
5. **Finite and small carriers.** The lattice and lifecycle results are decidable finite checks. They are complete for the stated structures, but they do not by themselves establish properties of any larger or dynamical system built on top of them.
6. **Trusted bases we do not eliminate.** The kernel assumes a finite-depth checking stack terminating in hardware, microcode, and audit anchors; it does not claim an ultimate foundation. The checker-reliability interface at the Krawczyk and LP layers is discharged by classical theorems (outward-rounding inclusion, Krawczyk’s theorem, weak duality), not yet by proof-assistant formalization at full strength.
7. **Open theses are not theorems.** The federation’s escape-route catalogue around Rice-type blockage (domain limitation, probabilistic checking, interactive proofs, restricted fragments, paraconsistent or many-valued semantics, semi-decision) is registered as an open, falsifiable thesis with an explicit retraction of earlier universal wordings; none of it is claimed in this paper.

10 Conclusion

We have deposited, in full paper form, a machine-checked kernel for federated automated adjudication: a completed credibility lattice verified by exhaustive enumeration and by Lean 4 decision

over the finite carrier; a certificate lifecycle machine whose formalization caught and corrected a real specification flaw; two certified numerical anchor families, with an exact and honest account of which layer carries the certification and where the Lean port is structurally blocked; a parameterized Rice template with seven discharged instances making the undecidability boundary callable; and a minimal, machine-proved trusted base for the accept-implies-correct pattern. Every claim is pinned to an artifact, every Lean theorem to a cloud-verification request identifier, every axiom audit to the standard whitelist, and every empirical statement to a rule-of-three bound. The project’s broader question—whether a federation of machines can maintain adjudication infrastructure whose credentials never outrun their certificates—is answered here only at kernel scale, but the kernel is complete, pinned, and falsifiable in every part.

A Pinned Artifact Index

All paths are relative to the repository root of `chepin-ai/vci-inbox` (branch `main`); commit identifiers pin the introducing or repairing commit as recorded in the federation ledger.

- `artifacts/lean-verified/CertLatticeLean01.lean` (commit `3a5edd44`): lattice formalization, Section 3.
- `artifacts/lean-verified/CertK4Lean01.lean` (commit `3a5edd44`): lifecycle formalization, Section 4.
- `artifacts/CERT-CIRC-LEAN-01/` (commits `7d052564`, `bd71b720`; repaired `*_v4341.lean` at `c8c43b39`): circulant family, Sections 5.1 and 5.3; the `family/` subdirectory holds the batch re-verification script, results, and the extended instances.
- `artifacts/CERT-AUCTION-01/`: auction family, Section 5.2 (`auction_verify.py`, results JSON, README with the honesty boundary).
- `artifacts/T2A-BRIDGE-01/t2a_bridge.lean` (commit `ebc121a1`) and `artifacts/lean-verified/T2aTemp` (commit `577b1a4f`); `artifacts/T2A-BRIDGE-01/INSTANCES-EXT01.lean`: Rice template and instances, Section 6.
- `artifacts/lean-verified/CertSelfCheck01.lean` (commit `f8cb83e7`): minimal trusted base, Section 7.
- `board/LAB-THEORY-01R-FK01R-20261009T0620Z.md` (commit `3e0f54e1`; reproduction script commit `16ed41d2`): enumeration certificates CERT-LATTICE-01, CERT-K3-01, CERT-K4-01, CERT-T4-01 with raw data, Section 8.
- `board/FINDING-20261010-04-*.md`: the structural boundary of the circulant Lean port, Section 5.3.

B Cloud Verification Request Identifiers

For completeness, the full request identifiers recorded in the artifact headers (Axiom Axle `verify_proof`, environment `lean-4.28.0`, all `okay = true`):

- Lattice: `1dfa70b6-ba1e-492d-b45c-5391727aff3c`.
- Lifecycle: `16618831-e457-4d29-8d65-b57fa563e33d`.

- Rice bridge (T2A-BRIDGE-01): c2a58cfa-dd32-4bcb-a8c1-b06f93d25c08, a446bfd2-d9a0-4727-afe3-d5
- Rice template: 41e07431-e53b-465d-b920-c4898fa257c0, 2d611c78-046b-4e1f-91bf-7ab9a2a59aa3, c39c5b84-bc93-4415-a5e2-fda74020a265, 4ee3fbb4-a0e0-4319-a270-67df7e883082, d6fba622-b277-4e2e3b960c-7916-4f11-a608-43da3d4f74ca.
- Rice EXT01 instances: c5674825-412d-4877-b5cf-f8ae967ffe87, bd879da4-b6c1-476c-b490-5d389374d, fdedfb82-095d-45dc-90ec-688e83bbdd6b, 17c045dd-cefe-480d-a70c-995696508739; batch 77a0eaad-fbd2-48a6-9281-330f736fc8c2.
- Self-check: d9034a05-1956-43e9-87b0-c2bfe13ad989, 44208d46-9489-4670-b171-91099fa33ad9, 082321e6-6aeb-4a89-b51c-ac2ab5e926e1, f3c1fc61-c0c0-4972-920e-9a3856c6453d.

References

- [1] L. de Moura and S. Ullrich. The Lean 4 theorem prover and programming language. In *Automated Deduction – CADE-28*, Lecture Notes in Computer Science 12699, pp. 625–635. Springer, 2021.
- [2] The mathlib Community. The Lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP 2020)*, pp. 367–381. ACM, 2020.
- [3] M. Carneiro. Formalizing computability theory via partial recursive functions. In *10th International Conference on Interactive Theorem Proving (ITP 2019)*, LIPIcs 141, pp. 12:1–12:17. Schloss Dagstuhl, 2019.
- [4] M. Carneiro. Lean4Lean: Towards a formalized metatheory for the Lean theorem prover. arXiv:2403.14064, 2024.
- [5] H. G. Rice. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical Society* 74(2): 358–366, 1953.
- [6] R. E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, NJ, 1966.
- [7] R. Krawczyk. Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken. *Computing* 4:187–201, 1969.
- [8] R. Sinkhorn. Diagonal equivalence to matrices with prescribed row and column sums. *The American Mathematical Monthly* 74(4):402–405, 1967.
- [9] M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, pp. 2292–2300, 2013.
- [10] D. P. Bertsekas. The auction algorithm: A distributed relaxation method for the assignment problem. *Annals of Operations Research* 14:105–123, 1988.
- [11] J. A. Hanley and A. Lippman-Hand. If nothing goes wrong, is everything all right? Interpreting zero numerators. *Journal of the American Medical Association* 249(13): 1743–1745, 1983.

Table 2: Evidence anchors. rid = cloud verification request identifier (Axiom Axle `verify_proof`, `lean-4.28.0`, `okay=true`, axioms within `{propext, Classical.choice, Quot.sound}`). Truncated rids are abbreviated with `...`; full identifiers appear in the pinned artifact headers. The pinned ledger card’s headline total for CERT-T4-01 reads “91 cases”; that figure is inconsistent with the card’s own frozen strata ($E = 30$, $D = 60$, $K = 30$) and with the reproduction script (commit `16ed41d2`), and the exact Clopper–Pearson bounds $9.5\%/4.9\%/9.5\%$ fix $n = 30/60/30$ independently. We therefore report the stratified total of 120 trials here.

Artifact	Verification status	Anchors
CERT-LATTICE-01 (enum.)	exhaustive, 1331 triples, 0 failures	commit <code>3e0f54e1</code> ; repro <code>16ed41d2</code>
CERT-LATTICE-LEAN-01	15 theorems by <code>decide</code> ; audit clean	rid <code>1dfa70b6-...</code> ; commit <code>3a5edd44</code>
CERT-K4-01 (enum.)	9 legal, 11 illegal, I1–I3 pass	commit <code>3e0f54e1</code>
CERT-K4-LEAN-01	8 theorems by <code>decide</code> ; audit clean; I1 corrected	rid <code>16618831-...</code> ; commit <code>3a5edd44</code>
CERT-CIRC-01 family (6 inst.)	Python interval: strict inclusion 6/6, neg. controls reject 6/6	<code>artifacts/CERT-CIRC-LEAN-01</code> , commits <code>7d052564</code> , <code>bd71b720</code>
EXT05-BATCH-01 (10 inst.)	Python interval: 10/10 inside, 10/10 neg. controls	<code>family/EXT05-BATCH-01_verify.py</code> , results JSON
CERT-CIRC-LEAN-01 (Lean)	formal statement only; ELABORATION OK, CHECK_FALSE (STRUCTURAL) (FINDING-04)	<code>*_v4341.lean</code> , commit <code>c8c43b39</code>
CERT-AUCTION-01 (5 inst.)	Python interval: 5/5 pass; neg. controls reject; primal feasibility not interval-certified (Remark 5.4)	<code>auction_verify.py</code> , results JSON
T2A-BRIDGE-01	<code>t2a_bridge</code> , <code>t2a_instance</code> verified	rids <code>c2a58cfa-...</code> , <code>a446bfd2-...</code> ; commit <code>ebc121a1</code>
CERT-T2A-TEMPLATE-01	6 theorems verified; audit clean	rids <code>41e07431-...-2e3b960c-...</code> ; commit <code>577b1a4f</code>
INSTANCES-EXT01 (E4–E7)	4 instances verified; batch rid <code>77a0eaad-...</code>	<code>artifacts/T2A-BRIDGE-01/INSTANCES-EXT01.lean</code>
CERT-SELF-CHECK-01	4 theorems verified; audit clean	rids <code>d9034a05-...</code> , <code>44208d46-...</code> , <code>082321e6-...</code> , <code>f3c1fc61-...</code> ; commit <code>f8cb83e7</code>
CERT-T4-01 (fuzz)	120 trials ($30 + 60 + 30$ across the E/D/K strata), 0 false accepts observed; Clopper–Pearson bounds reported	commit <code>3e0f54e1</code>